DEARVR

UNITY

PERFECT SPATIALIZATION

FOR UNITY

DEARREALITY

DEARVR

UNITY

USER MANUAL
v1.6

Please read this manual carefully before using the software.

Using headphones requires responsible listening!

Last updated: May 2020

Quick Start Guide

1. **Import**

   - Import the **dearVR Asset** to your project.
     For details on how to import a Unity package, refer to chapter 2.2.

2. **Setup**

   - In Unity **AudioManager,** set **Spatializer** Plugin to **dearVR Engine**.
     Go to menu bar: Edit -> Project Setting -> Audio -> Spatializer Plugin.

3. **Listen**

   - Run the **dearVR Demo Scene**.

   Alternatively:

   - Add **`DearVR Manager`** component to a game object in your scene.
   - Create a new game object and add the **`DearVR Source`** component.

   Note: In the Demo Scene, move with WASD and rotate with mouse down. You can change a room preset or any other parameter on a `DearVR Source` anytime. Press keys **"R"** and **"T"** to switch presets and keys **"F"** and **"G"** to switch audio clip.

Using dearVR Reverb Sends

To illustrate how to use dearVR Reverb Sends enable **`dearVRSource_Send`** object (and disable `dearVRSource_internal`) in the dearVR Demo Scene.

1. Create an Audio Mixer and add a new group (e.g., name it Reverb Bus 1).

2. Right-click on that group, **add effect** at the top and select **dearVR Reverb**.
   Please repeat Steps 1. & 2. for more groups and Reverb Plugin instances (i.e., using multiple Reverb Groups for different room presets).

3. Select the **dearVR Reverb Plugin**, choose a room preset (in the Inspector Window), and set the **Reverb ID** between 1 and 100!
   IMPORTANT: Each dearVR Reverb Plugin needs a unique Reverb ID!

4. For each **dearVR Source,** set **INTERNAL REVERB** to OFF. Use **SIZE** to set the numbers (size) of **dearVR Reverb Plugins** you want to address with the audio source.

   Note: Select the same room preset for a source object as for the main Reverb Plugin to use the corresponding early reflections.

5. Set Reverb ID(s) and Send-Level(s). The Reverb ID(s) determines the Reverb Plugins; the selected source object sends the signal.

   Note: SEND determines the individual Send Level to a Reverb Plugin while **REVERB LEVEL** (in the LEVELS Section) acts as a Master Send Fader for all Reverb Groups (if using multiple Reverb Groups).

6. For each **dearVR Source,** set the **AudioSources Output** to **Audio Mixer Master**.

   WARNING: Do not send the AudioSources output to the Reverb Bus!
   Otherwise, the binaural signal gets processed by the dearVR Reverb again!

WARNING: UPDATING EXISTING PROJECTS FROM EARLIER THAN 1.5.0 TO v1.5.1
We replaced our closed managed plugin DLLs with the open C#-Source files in our plugin. Due to this change, upgrading a project from earlier versions to v1.5.1 needs some more adjustments than just importing the new unity package. Please see section 7 for more information on how to upgrade your project.

Table of Contents

1 Introduction

Thank you for purchasing our dearVR PRO Plugin and welcome to the next step of immersive audio production. With the dearVR 3D audio technology, you can design fantastic new music mixes for headphones or create deep auditory worlds and sound design within your DAW.
This manual helps you understand the dearVR PRO Plugin and how to use it in your projects.

Important Note:
dearVR is a 3D audio technology for headphones. Any kind or brand will do, but if you set the output format to “Binaural”, you have to use a headphone for the plugin to work correctly. Please check that your left and right earpieces are suitably placed, and let’s get started!

Have fun!

1.1 About binaural 3D Audio

Binaural 3D Audio is a technology that simulates the human spatial hearing via headphones. If you listen to common stereo audio with headphones, the perception of all sound sources is located inside your head - between your left and your right ear. With 3D Audio, you get the sound outside of your head where it belongs. A sound appears to emanate from a specific point - anywhere within a full 360° three-dimensional sphere.

This perception gives you the ability to position a sound object all around the listener - behind, in front of, to the right or the left of and even below or above.

The quality of a 3D Audio rendering process depends on many factors - primarily the shape of our body, our head, and our ears. That’s why a mix with 3D Audio can sound different to different people. Our uniqueness as human beings is a limitation for practical 3D Audio technology.

Another typical problem that occurs with binaural 3D Audio is the front-back confusion. Our natural hearing uses small micro-movements of our head to optimize the localization of a sound source. The head tracking technology, which is part of virtual reality devices, offers a solution to this problem.

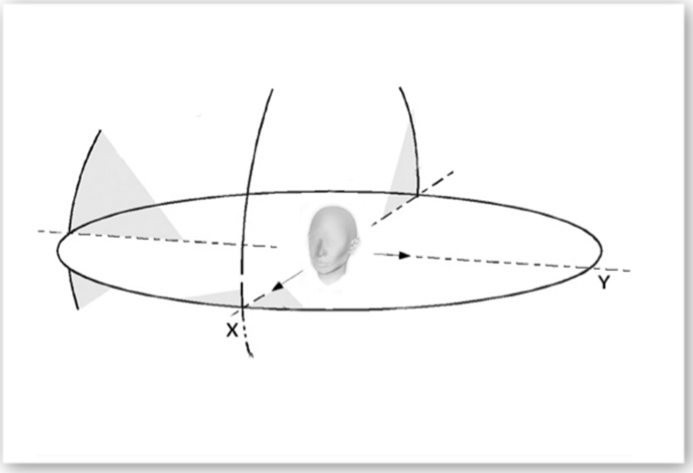


Illustration 1.1 - Full 360-degree 3D audio sphere

Binaural 3D Audio is not made for listening via stereo speakers. Although, in general, the playback is possible, you face strong colorations of the sound depending on your playback system. The reason for this is crosstalk, meaning a large portion of the left speaker signal is going to the right ear of the listener. Similarly, a large portion of the right speaker signal is progressing to the left ear of the listener. With headphones, this is different: both the left and the right channel signal reaches its respective ear.

1.2 dearVR audio reality engine

We call our dearVR technology an audio reality engine because it can produce stunningly realistic auditory worlds, comparable to our natural listening. For this, we listened back to our modelling technology over and over again and tweaked every parameter with our ears.

The acoustic modeling of an environment needs more than just a 3D spatial location. It combines distance, motion, reflections, and reverb to complete a simulation of an acoustic scene.
All these phenomena you can now use for your sound design with just one plugin:
The dearVR audio reality engine!

2 Installation

This chapter describes how to set up your dearVR Unity Asset. It assumes that you are using Unity's built-in audio engine. If you are using a third-party audio middleware like FMOD Studio or Wwise, please refer to additional information on our website.

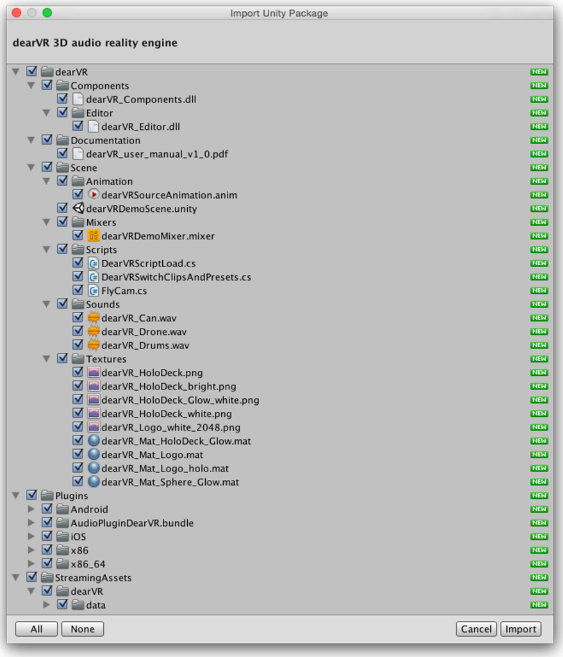
2.1 Requirements

- Unity 5.2 or higher.

- The `dearVR.unitypackage` from the Asset Store.

2.2 Importing dearVR

Import the complete dearVR Plugin via the Asset Store into your project.



Import Unity Package

dearVR 3D audio reality engine

```
▼ ☑ dearVR
  ▼ ☑ Components
      ☑ dearVR_Components.dll
    ▼ ☑ Editor
        ☑ dearVR_Editor.dll
  ▼ ☑ Documentation
      ☑ dearVR_user_manual_v1_0.pdf
  ▼ ☑ Scene
    ▼ ☑ Animation
        ☑ dearVRSourceAnimation.anim
      ☑ dearVRDemoScene.unity
    ▼ ☑ Mixers
        ☑ dearVRDemoMixer.mixer
    ▼ ☑ Scripts
        ☑ DearVRScriptLoad.cs
        ☑ DearVRSwitchClipsAndPresets.cs
        ☑ FlyCam.cs
    ▼ ☑ Sounds
        ☑ dearVR_Can.wav
        ☑ dearVR_Drone.wav
        ☑ dearVR_Drums.wav
    ▼ ☑ Textures
        ☑ dearVR_HoloDeck.png
        ☑ dearVR_HoloDeck_bright.png
        ☑ dearVR_HoloDeck_Glow_white.png
        ☑ dearVR_HoloDeck_white.png
        ☑ dearVR_Logo_white_2048.png
        ☑ dearVR_Mat_HoloDeck_Glow.mat
        ☑ dearVR_Mat_Logo.mat
        ☑ dearVR_Mat_Logo_holo.mat
        ☑ dearVR_Mat_Sphere_Glow.mat
▼ ☑ Plugins
  ▶ ☑ Android
  ▶ ☑ AudioPluginDearVR.bundle
  ▶ ☑ iOS
  ▶ ☑ x86
  ▶ ☑ x86_64
▼ ☑ StreamingAssets
  ▼ ☑ dearVR
    ▶ ☑ data
```

Illustration 2.1 - Import Asset

2.3 The dearVR folders

After import three new folders appear inside your project:

- `dearVR`
- `Plugins`
- `StreamingAssets`

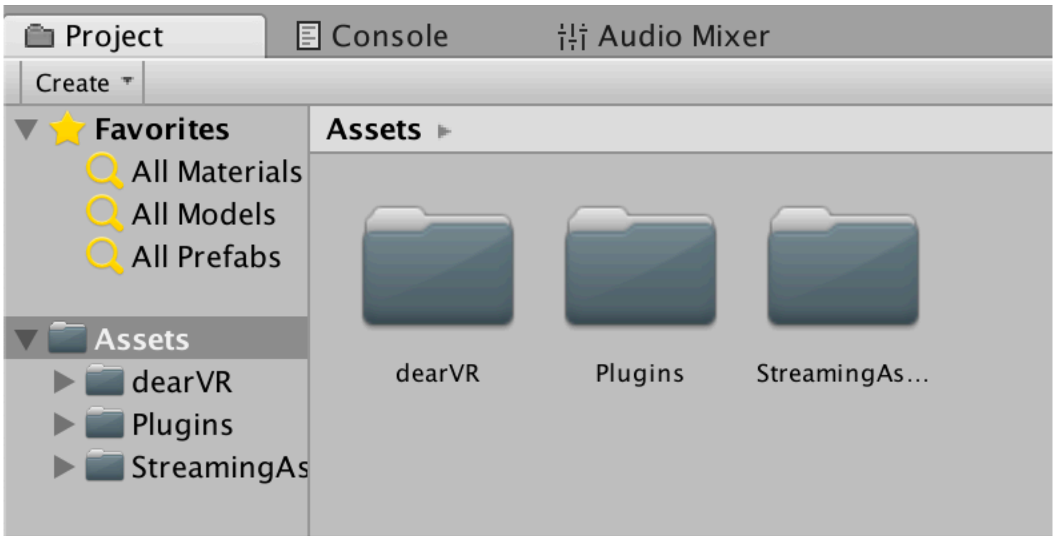


Illustration 2.2 - Imported folders in the Project Window

The dearVR folder contains:

a) dearVR Demo - demo scene to get a first impression of the Plugin.

b) Components - `dearVR Manager` and `dearVR Source`.

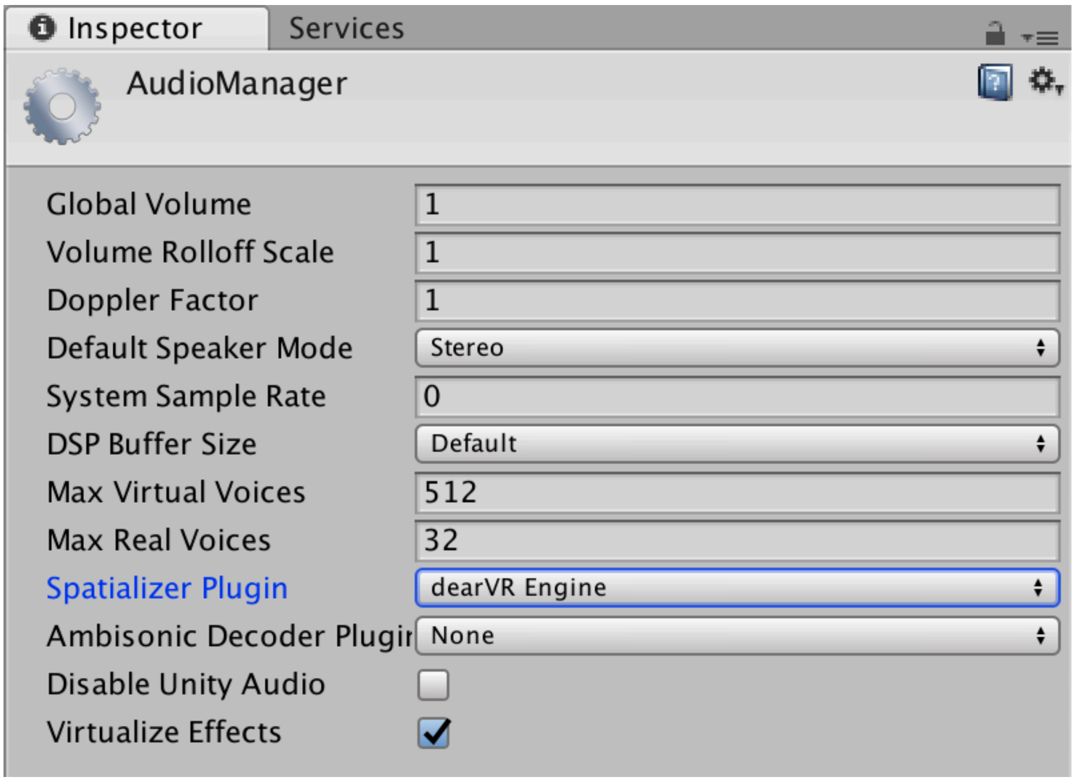
Note: `dearVR_Editor` is an internal system component.

The `Plugins` folder contains the native plugins for OSX, Windows, iOS, and Android, while the `StreamingAssets` folder contains necessary data files for the dearVR Engine.

2.4 Choose dearVR as your default Spatializer Plugin

The dearVR Engine uses the Unity Spatializer SDK - introduced with Unity 5.2 as an extension of the native audio SDK and built for third-party 3D Audio solutions like dearVR. To use the plugin, you have to set it as the default tool for spatialization in Unity.

- Go to menu bar: Edit -> Project Setting -> Audio -> Spatializer Plugin.
- Within the Unity Audio Manager, you find the Spatializer Plugin Menu.
- Choose dearVR Engine.



Global Volume	1
Volume Rolloff Scale	1
Doppler Factor	1
Default Speaker Mode	Stereo
System Sample Rate	0
DSP Buffer Size	Default
Max Virtual Voices	512
Max Real Voices	32
Spatializer Plugin	dearVR Engine
Ambisonic Decoder Plugir	None
Disable Unity Audio	☐
Virtualize Effects	✓

Illustration 2.3 - Unity Audio Manager Spatializer Plugin

Note:

- 24000 Hz / 44100 Hz / 48000 Hz Sample Rate are supported.
- The performance load for spatialization depends on DSP Buffer Size Settings.

2.5 Getting Started

How to configure a new scene:

1. Add the `DearVR Manager` component to a game object in your scene.
2. Create a new game object and add the `DearVR Source` component. If the game object doesn't have an `AudioSource` yet, a new one is created automatically.

Done - that's all!

3 Overview of dearVR Engine

The dearVR Unity Asset is an audio reality processor, enabling you to virtualize many different kinds of acoustic settings within your game with a vast amount of realism. The virtualization of an acoustic environment with object-based sound sources refers to combining 3D spatial location cues with distance, motion, and ambiance. All these parts are needed to achieve a realistic simulation of an acoustic scene - far beyond simple 3D positioning. For this reason, the dearVR Engine combines object-based 3D positioning with real-time generated reflections depending on the room positioning and late diffuse reverberation.

The dearVR Plugin contains the following processing units:

1. Position
2. Reflections
3. Reverb

3.1 Position

The positional processing renders distance, listening angle, and elevation relative to the listener. Using the values provided by the Unity game engine, it converts a mono audio track into a positional 3D Sound Object.

3.2 Reflections

A signal being reflected once or twice from parts of listening space - walls, ceilings, and floor - arrives shortly after the direct signal at the listener's position. Such early reflections can be seen as a transition period before the reverberant field has built up.

First reflections are responsible for our impression of the general character and the size of a room. In binaural rendering, they are of great importance for a realistic localization and the impression of sounds coming from outside of the head.

You can model early reflections by considering acoustic boundaries as acoustic mirrors. Depending on the listener's position in a room, and his distance to the boundaries, the time a reflection takes to arrive at his ears varies. Reflections also vary depending on the position of the sound source in the room.

The dearVR auralization generates reflection patterns depending on the listener's position and the sound source position. If the listener or a sound object moves, the engine recalculates the reflections' properties in real-time.

3.3 Reverb

Reverb itself is an extremely complex reflection and diffusion pattern that builds up to a dense thickness from the moment you hear the original dry sound. For the dearVR Engine, we captured the acoustic characteristics of different rooms and locations and created over 40 room presets. You can quickly adapt these room characteristics to a specific scene by changing the size or applying a reverb filter.

4 dearVR Components

In this chapter, we take a look at the three main components you find inside the dearVR Asset and their corresponding parameters. You can change any parameter in real-time during playback.

1. `dearVR Source` component.
Add this component to every Audio Object in your scene.

2. `dearVR Manager` component.
One instance of the `dearVR Manager` must exist in every scene.

3. `dearVR Reverb` Plugin
Add this native audio plugin in a mixer group to create a spatialization reverb bus.

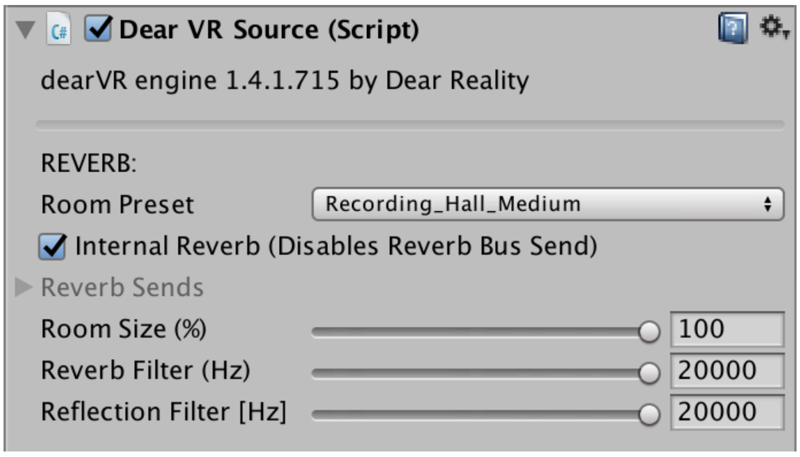
4.1 dearVR Source

For binaural processing, each Unity `AudioSource` requires the `dearVR Source` component. Once you have added the component to a game object, it generates an audio source component automatically if it does not already exist.

There are five main parameter sections within the inspector window of a `dearVR Source` component: Reverb, Levels, Settings, Occlusion, and Obstruction.

Tip
Don’t mix binaural with non-binaural audio! If you combine conventional stereo or mono audio playback with 3D audio objects, you wreck immersion and spatialization.

4.1.1 Reverb


Dear VR Source (Script)

dearVR engine 1.4.1.715 by Dear Reality

REVERB:

Room Preset: Recording_Hall_Medium

✓ Internal Reverb (Disables Reverb Bus Send)

Reverb Sends

Room Size (%): 100

Reverb Filter (Hz): 20000

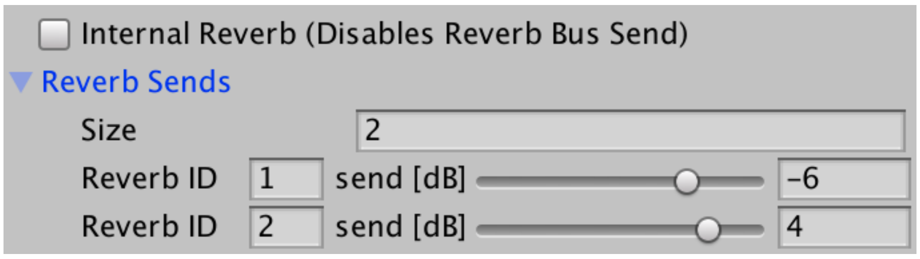
Reflection Filter [Hz]: 20000

Illustration 4.1 - Reverb parameters

Room Preset	Menu to select different room presets. (Refer to 5.6 Preset List) **Note:** The more extensive a room, the longer the reverb – and the more processing power needed!
Internal Reverb	Select whether to use internal reverb processing on the Audio Source or a Reverb Send to address a `dearVR Reverb` plugin instance in the Unity Audio Mixer. **Default:** ON **Note:** Internal Reverb processing needs far more processing power.
Reverb Filter	A low-pass filter applied to the reverb signal. Set frequency values between 500 Hz to 19999 Hz. The default slider position is 20000 Hz which we use for an optimized internal value for the reverb filter!
Room Size	Factor to decrease room size. Values (50% ⇔ 100%) **Note:** Parameter is disabled if using Reverb Groups.
Reflection Filter	A low-pass filter applied to the reflection signal—Set frequency values between 500Hz to 19999 Hz. The default slider position is 20000 Hz, which we use for an optimized internal value for the reverb filter!

Reverb Sends:

We recommend using reverb sends for each audio source. This way, various audio sources with different positions and reflections can share the same reverb to reduce processing load. Reverb Sends also enable crossfading between different reverbs – e.g., needed for sound sources moving from one room to another.



☐ Internal Reverb (Disables Reverb Bus Send)

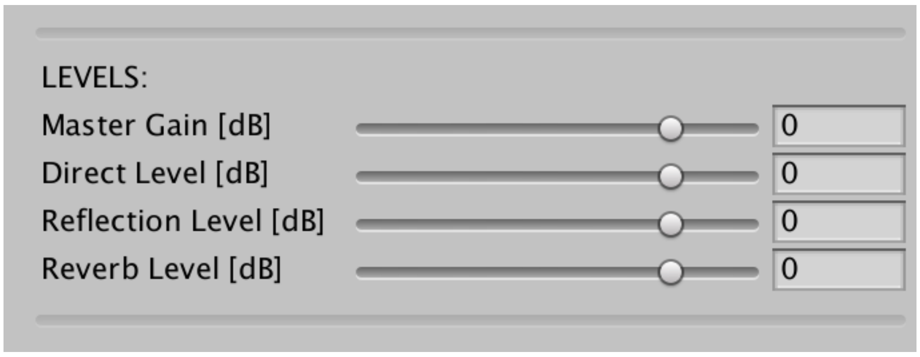
▼ Reverb Sends

Size	2		
Reverb ID	1	send [dB]	-6
Reverb ID	2	send [dB]	4

Illustration 4.2 - Reverb sends

Internal Reverb	OFF
Reverb Sends Size	Sets the number of Reverb Sends.
Reverb ID	Enter a Reverb ID for each `dearVR Reverb` Plugin you want to target with the Reverb Send. (You have to define a unique Reverb ID in each `dearVR Reverb` Plugin instance). **Note:** Create a `dearVR Reverb` Plugin in a Mixer Group within the Unity Audio Mixer.
Send Level	Sets the audio signal level send internally to the `dearVR Reverb` Plugin. **Note:** All Reverb Send levels are affected by the Reverb Level Fader in the Levels section (refer to 4.1.2).

4.1.2 Levels


LEVELS:

Master Gain [dB] 0

Direct Level [dB] 0

Reflection Level [dB] 0

Reverb Level [dB] 0

Illustration 4.3 - Levels

Master Gain	Sets the overall output gain (-96dB ⇔ +24dB)
Direct Level	Sets the level of the audio source direct signal - independent of reflections and reverb. (-96dB ⇔ +24dB)
Reflection Level	Sets the overall gain of the reflections. (-96dB ⇔ +24dB)
Reverb Level	Sets the overall gain of the reverb signal. (-96dB ⇔ +24dB) **Note:** This level affects all Reverb Sends

Tip
Values for direct, reflection, and reverb levels influence the distance perception.

4.1.3 Settings

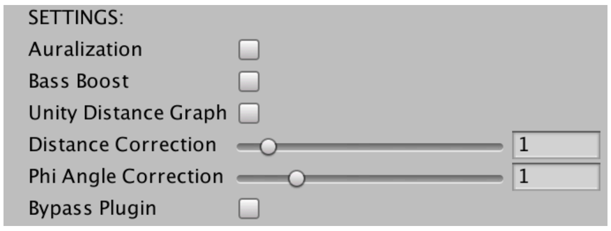
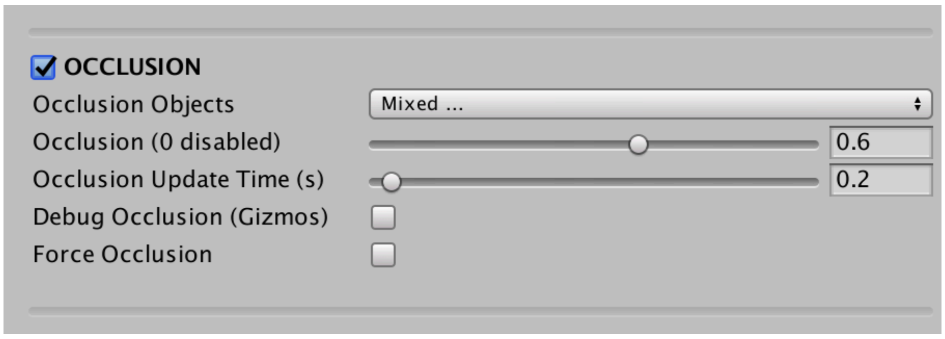

SETTINGS:
Auralization ☐
Bass Boost ☐
Unity Distance Graph ☐
Distance Correction 1
Phi Angle Correction 1
Bypass Plugin ☐

Illustration 4.4 - Settings

Auralization	Activates real-time processing to generate first reflections corresponding to the listener and sound object position relative to wall distances and reflection boundaries. **Note:** If room geometry is not set manually or analyzed with the real-time room analyzer (for details, see chapter 4.2), the operating room preset determines the geometry.
Bass Boost	Enable bass enhancement (On / Off) **Default:** OFF
Unity Distance Graph	Activates Unity Distance Attenuation in the Audio Source 3D Sound Settings (refer to 4.1.7) **Default:** OFF **Note:** The maximum distance value for the internal dearVR processing is set to 28 meters. You can modify this value by using the distance correction parameter.
Distance Correction	Factor to scale the internal dearVR distance processing according to the given distance in the unity scene (0.01 ⇔ 10.0). **Default:** 1.0 **Note:** Value > 1.0 leads to increased distance perception. Value < 1.0 leads to decreased distance perception.
Phi Angle Correction	Factor to scale the internal dearVR horizontal-angle processing, according to the given phi angle in the unity scene (0.2 ⇔ 4.0) **Default:** 1.0 **Note:** Value < 1.0 results in more sideways position perception. Value > 1.0 results in more centred position perception.
Bypass Plugin	Bypass dearVR processing.

4.1.4 Occlusion

Acoustic occlusion describes the alteration within a sound field if the sonic wave is completely blocked, for example, by a wall, a closed window or door. With the parameter occlusion level, you can simulate the acoustic insulation of a wall, window, or door. Occlusion alters direct signal and reverberation signal of an audio source and is always measured between the audio listener and the audio source.



OCCLUSION [✓]

Occlusion Objects	Mixed ...
Occlusion (0 disabled)	0.6
Occlusion Update Time (s)	0.2
Debug Occlusion (Gizmos)	☐
Force Occlusion	☐

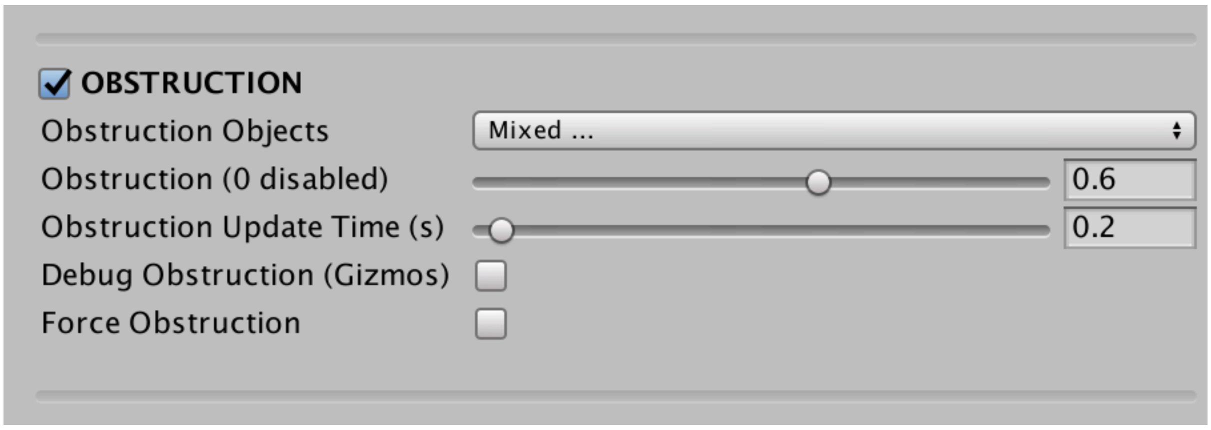
Illustration 4.5 - Occlusion

Occlusion Objects	Select the layer of objects that lead to occlusion.
Occlusion	Sets the level of occlusion if a sound source is occluded and blocked by wall, window, or door (0.0 ⇔ 1.0). **Default:** 0.6
Occlusion Update Time	Set the time (sec.) between updating occlusion detection (0.1 s ⇔ 5.0 s). **Default:** 0.2 s **Note:** Smaller values lead to increased performance load!
Debug Occlusion (Gizmos)	Visualizes ray casting between listener and sound source in the scene window. (green ray => no occlusion) (red ray => occlusion)
Force Occlusion	Occlusion is always processed.

4.1.5 Obstruction

Acoustic obstruction describes the alteration within a sound field if the direct sonic wave of a sound source is blocked by another object. Unlike occlusion, the obstructed sound source is in the same room with the listener.

Obstruction mainly influences the direct signal of an audio source. The reverberation signal gets less influenced if an object is obstructed. Obstruction is always measured between the audio listener and an audio source.



☑ OBSTRUCTION

Obstruction Objects	Mixed ...
Obstruction (0 disabled)	0.6
Obstruction Update Time (s)	0.2
Debug Obstruction (Gizmos)	☐
Force Obstruction	☐

Illustration 4.6 - Obstruction

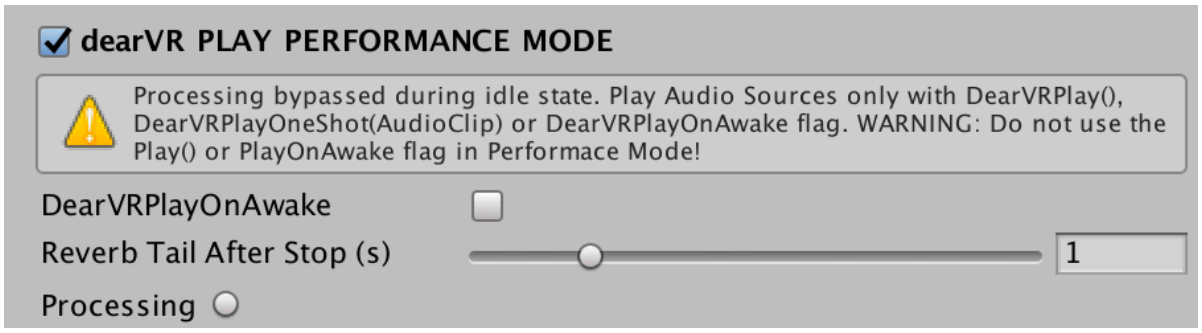
Obstruction Objects	Select the layer of objects that lead to obstruction.
Obstruction	It sets the level of obstruction if another object obstructs the sound source (0.0 ⇔ 1.0). **Default:** 0.6
Obstruction Update Time	Set the time (sec.) between updating obstruction detection (0.1 s ⇔ 5.0 s). **Default:** 0.2 s **Note:** Smaller values lead to increased performance load!
Debug Obstruction (Gizmos)	Visualizes ray casting between listener and sound source in the scene window. (green ray => no obstruction) (red ray => obstruction)
Force Obstruction	Obstruction is always processed.

4.1.6 Performance Mode

The Performance Mode bypasses the processing of spatialized audio sources if they are in an idle state. Usually, all audio sources with the Spatialize checkbox enabled are processed by Unity, no matter if they are playing or not. The Performance Mode is an important feature to avoid unnecessary processing load.

Important: In Performance Mode play audio sources only with `DearVRPlay()` or `DearVRPlayOneShot (AudioClip)` or `DearVRPlayOnAwake` flag.

WARNING: Do not use `Play()` or `PlayOnAwake()` flag in Performance Mode!


☑ dearVR PLAY PERFORMANCE MODE

Processing bypassed during idle state. Play Audio Sources only with DearVRPlay(), DearVRPlayOneShot(AudioClip) or DearVRPlayOnAwake flag. WARNING: Do not use the Play() or PlayOnAwake flag in Performace Mode!

DearVRPlayOnAwake ☐

Reverb Tail After Stop (s) 1

Processing

Illustration 4.7 - Performance mode

dearVR Play Performance Mode	Activates Performance Mode. Spatial audio sources are NOT processed during the idle state. **Note:** Unity usually processes all audio sources with spatialization on – even if they aren't playing.
DearVRPlayOnAwake	`PlayOnAwake` flag for the Performance Mode. **Note:** Never activate the audio sources `PlayOnAwake` in Performance Mode.
Reverb Tail After Stop	Set length (sec.) of processed Reverb Tail after audio source stop. **Note:** Only important if internal Reverb is active.
Processing Indicator	When lit green, the processing is active

4.1.7 Audio Source

Some settings of the Audio Source component also affect the dearVR Settings. Not mentioned settings act as usual in Unity.
For more information about Unity Audio Source settings refer to the Unity manual:

http://docs.unity3d.com/Manual/class-AudioSource.html

Spatialize	Automatically set to ON if a `dearVR Source` component is added.
Bypass Effects / Listener Effects / Reverb Zones	These Settings do not have any effect on the dearVR rendering.
Stereo Pan	No effect
Spatial Blend	Automatically set to 2D if a `dearVR Source` component is added. **Important:** Keep parameter at 2D!
Reverb Zone Mix	If Unity Distance Graph in dearVR Source enabled, slider, or graph control Reverb Level.
Doppler Level	No effect
Spread	No effect
3D Sound Settings Graph	The **volume** graph alters the Direct Level over distance. **Reverb Zone** graph alters the Reverb Level over distance. **Note:** If the distance is farther than the **Max Distance** value, the farthest value is processed. Set **Volume** and **ReverbZone** to zero at **Max Distance** to mute both for higher distances.

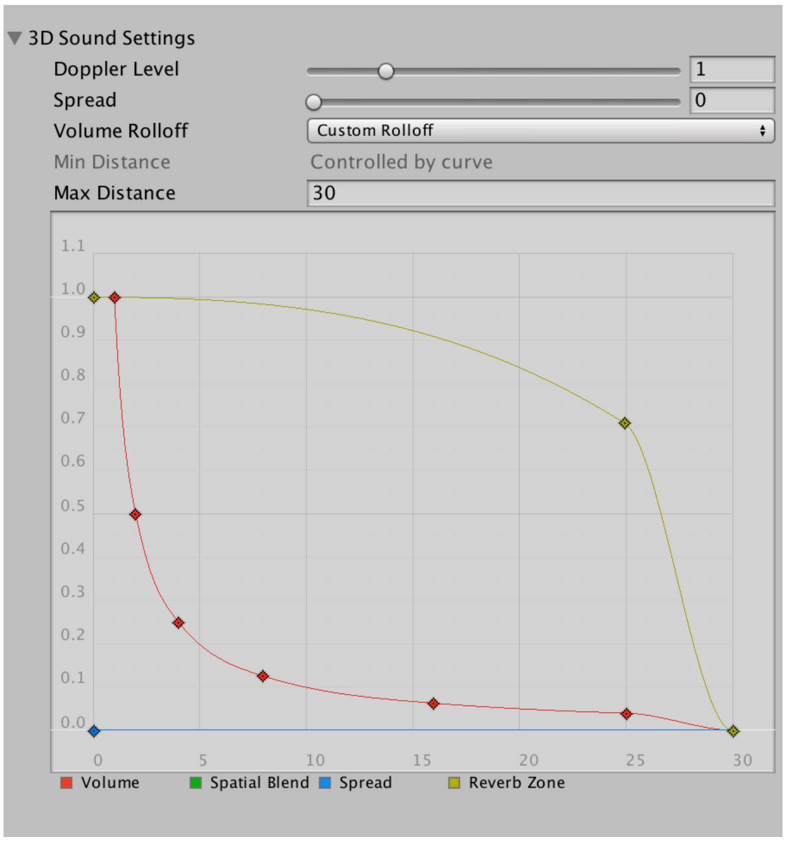

▼ 3D Sound Settings
Doppler Level 1
Spread 0
Volume Rolloff Custom Rolloff
Min Distance Controlled by curve
Max Distance 30

Illustration 4.8 - Audio Source Graph

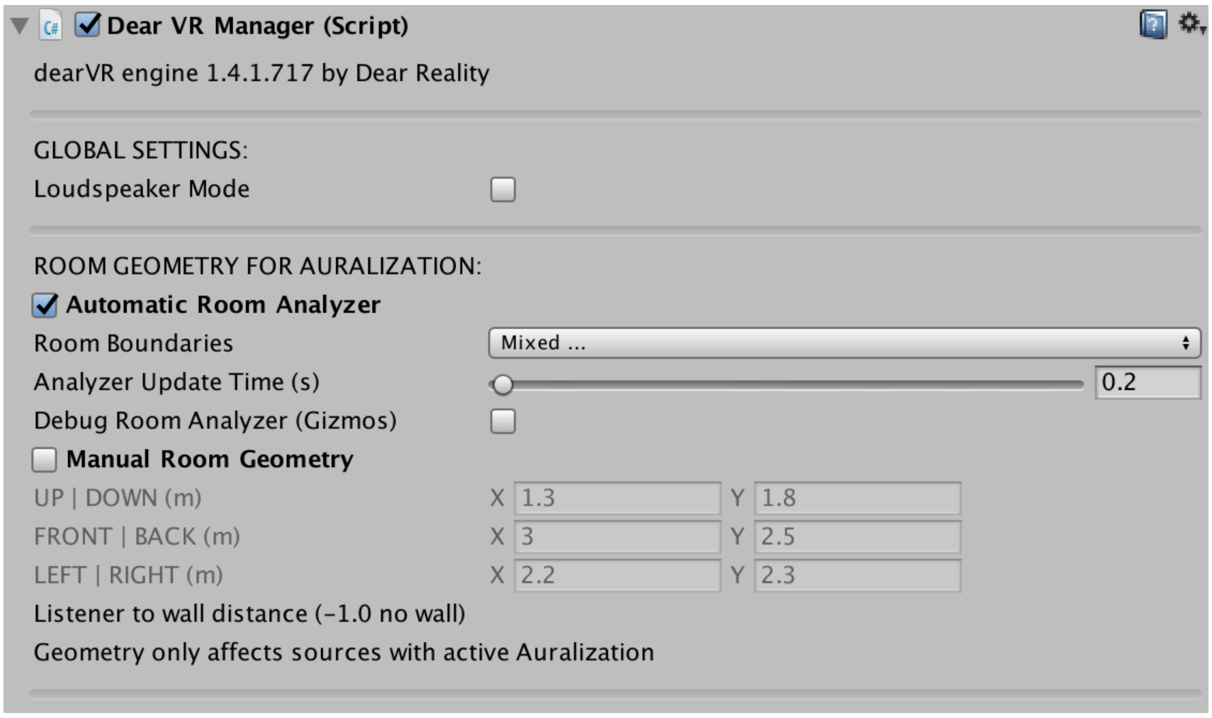
4.2 dearVR Manager

The `dearVR Manager` component is mandatory in each scene or a global root scene. It defines global settings for the dearVR Engine.

Note: For details on how to use a global scene, please refer to the unity manual.

http://docs.unity3d.com/Manual/MultiSceneEditing.html

Within the `dearVR Manager`, you can set the main parameters for auralization and the room analyzer.


Dear VR Manager (Script)

dearVR engine 1.4.1.717 by Dear Reality

GLOBAL SETTINGS:

Loudspeaker Mode ☐

ROOM GEOMETRY FOR AURALIZATION:

☑ Automatic Room Analyzer

Room Boundaries: Mixed ...

Analyzer Update Time (s): 0.2

Debug Room Analyzer (Gizmos) ☐

☐ Manual Room Geometry

UP | DOWN (m)	X 1.3	Y 1.8
FRONT | BACK (m)	X 3	Y 2.5
LEFT | RIGHT (m)	X 2.2	Y 2.3

Listener to wall distance (-1.0 no wall)

Geometry only affects sources with active Auralization

Illustration 4.9 - DearVR Manager

The dearVR Manager's settings are globally stored in an asset file under `Assets/dearVR/Ressources/DearVRManagerState.asset`, but you can also put it somewhere else as long as it is inside the `Assets`-folder in a folder called `Resources`. This also means that every dearVR Manager in every scene has the same state. When you change the settings in one scene, it is automatically transferred to all of them.

WARNING: Legacy issue

When you update a scene with a dearVR manager from a previous version to v1.5 / v1.5.1, it causes a loss of the dearVR manger's settings.
When updating a project, the dearVR manager needs to initialize a new DearVRManagerState.asset file. You can do this by changing a value in a dearVR manager instance, pushing the play button, or by creating one via the Assets menu.

Loudspeaker Mode	Bypass the binaural rendering for all sources. Reflections and Reverb processing are still active. **Note:** Loudspeaker Mode enables you to switch from 3D Audio for headphones to a loudspeaker compatible mix. Distance attenuation, Reflection, and Reverb levels are still maintained.
Automatic Room Analyzer	Enable the real-time room analyzer to get scene geometry for auralization. Detected values are shown in Manual Room Geometry input boxes. **Note:** Without room analyzer or Manual Room Geometry active, reflections are generated based on fixed values fitting the room preset.
Room Boundaries	Select which objects are analyzed as room boundaries.
Analyzer Update Time	Set the Update-time (in seconds) between two ray casts. (0.1 s ⇔ 10.0 s) **Note:** Faster Update-time leads to more performance load.
Debug Room Analyser (Gizmos)	Visualize ray casting for room analyzer. Visible in Scene Window. **Note:** Use to control the objects detected as room boundaries.
Manual Room Geometry	Enable to set Room Geometry manually.
UP / DOWN	Listener’s distance to the ceiling / ground in meter.
FRONT / BACK	Listener’s distance to the front / rear wall measured in meter. Value modifies delay and level of reflection from front / rear direction.
LEFT / RIGHT	Listener’s distance to the left or right wall measured in meter. Value modifies delay and level of reflection from the left / right direction.

4.3 dearVR Reverb (Unity Audio Mixer Plugin)

Reverb sends allow various audio sources with different positions and reflections to share the same reverb. The signal of an audio source is sent to `dearVR Reverb` plugin instances within the Unity Audio Mixer. This way, the processing load is reduced by a multiple.

- Create an Audio Mixer and add a new group (e.g., name it Reverb Bus 1).
- Right-click on that group, add effect at the top and select dearVR Reverb.
- Select the dearVR Reverb Plugin and set the Reverb ID between 1 and 100. **IMPORTANT:** Each dearVR Reverb Plugin needs a unique Reverb ID!
- Choose the room preset and adjust the reverb parameter.

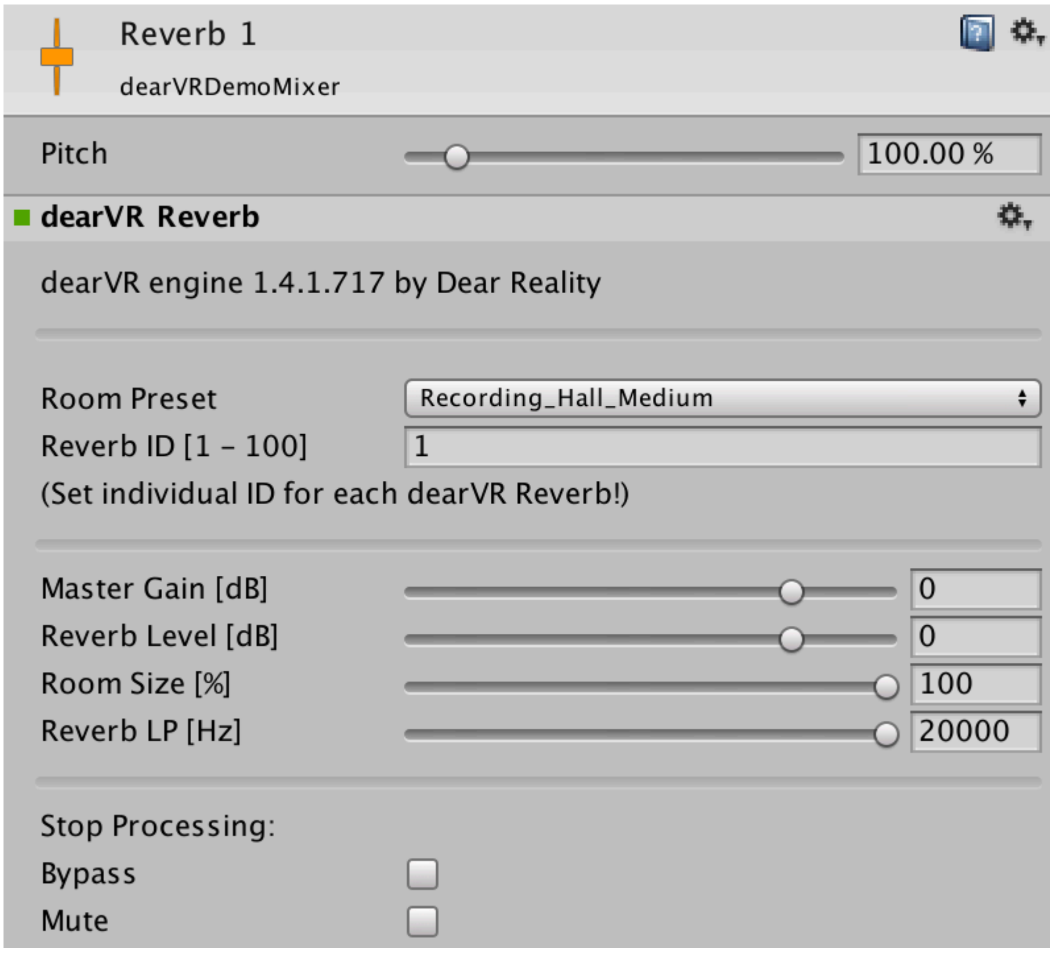

Reverb 1
dearVRDemoMixer
Pitch 100.00 %
dearVR Reverb
dearVR engine 1.4.1.717 by Dear Reality
Room Preset: Recording_Hall_Medium
Reverb ID [1 – 100]: 1
(Set individual ID for each dearVR Reverb!)
Master Gain [dB]: 0
Reverb Level [dB]: 0
Room Size [%]: 100
Reverb LP [Hz]: 20000
Stop Processing:
Bypass ☐
Mute ☐

Illustration 4.10 - dearVR Reverb Audio Mixer Plugin

Room Preset	Menu to select a Reverb Room preset. (Refer to chapter 5 for Preset List)
Reverb ID	Set an ID for the Reverb Group **IMPORTANT:** Each `dearVR Reverb` Plugin needs a unique Reverb ID!
Gain	Sets the output gain of the Reverb Plugin. **Note:** Slider is Pre-Fader Reverb Group in Audio Mixer.
Reverb	Sets the overall gain of the reverb signal. (-96dB ⇔ +24dB)
Reverb LP	A low-pass filter applied to the reverb signal. It sets frequency values between 500 Hz to 20000 Hz. **Note:** Default values are set within each room preset.
Room Size	Factor to decrease the Room Size. Values (50% ⇔ 100%)
Reverb Bass Boost	Enable bass enhancement for a reverb (On / Off) **Default:** OFF
Bypass	Bypass the processing to control reverb bus input.
Mute	Stop the processing to safe performance.

5 Preset List

Note: The more extensive a room, the longer the reverb – and the more performance is needed!

The Performance Level (1-10) illustrates the performance load. Level 10 needs the most performance.

A.	Rooms & Halls	Performance Level
1.	Concert Hall 1	5
2.	Concert Hall 2	5
3.	Recording Hall Large	7
4.	Recording Hall Small	4
5.	Kings Hall	5
6.	Cathedral	10
7.	Church	5
8.	Chapel	8
9.	Room Large	5
10.	Room Medium	4
11.	Room Small	3

B.	**Postproduction**	**Performance Level**
11.	Office 1	2
12.	Office 2	2
13.	Studio Small	2
14.	Conference Hall Small	3
15.	Cellar	4
16.	Empty Room	3
17.	Living Room Small	2
18.	Staircase	4
19	Corridor	5
20.	Bathroom	3
21.	Restroom	4
22.	Car 1	1
23.	Car 2	1
24.	Booth	2
25.	Cinema	2
26.	Warehouse	10
27.	Outdoor Street	5
28.	Outdoor Alley	5

C.	**Music production**	**Performance Level**
29.	Live Studio Room	5
30.	Live Stage	7
31.	Live Arena	7
32.	Ambience Heavy	4
33.	Ambience Plate	3
34.	Ambience Medium	3
35.	Ambience Small	2
36.	Vocal Hall 1	7
37.	Vocal Hall 2	7
38.	Vocal Plate	8
39.	Drum Room 1	6
40.	Drum Room 2	6
41.	Percussion Plate	5
42.	Percussion Ambience	2
43.	Acoustic Room	6
44.	String Hall	8
45.	String Plate	8

6 Building with dearVR Unity

Windows:

Windows builds require a Visual Studio 2015 runtime (or later) on the machine which runs the application.

iOS & Android:

1. The default sample rate on mobile is 24000 Hz.
You may change the sample rate to 44100 Hz or 48000 Hz for higher quality in the trade of performance.
To change the sample rate, go to menu bar Edit -> Project Setting -> Audio

2. The best practice is to use Reverb Sends and set DSP Buffer Size in the Audio Manager to Best Performance.

iOS:

In your XCode-Project open `/Classes/UnityAppController.mm`.

1. Add the line:

```
#include "../Libraries/Plugins/iOS/AudioPluginInterface.h"
```

2. In the same file replace line:

```
- (void)preStartUnity {}
```

with

```
- (void)preStartUnity {
UnityRegisterAudioPlugin(&UnityGetAudioEffectDefinitions ) ; }
```

7 Upgrading from earlier versions to v1.5.1

We replaced our closed managed plugin DLLs with the open C#-Source files in our plugin. Due to this change, upgrading a project from earlier versions than v.1.5.0 to v1.5.1 needs some more adjustments than just importing the new unity package.

7.1 Upgrading on macOS

- Delete the following files from your Unity project:
    - `/Assets/dearVR/Components/dearVR_Components.dll`
    - `/Assets/dearVR/Components/dearVR_Components.xml`
    - `/Assets/dearVR/Components/Editor/dearVR_Editor.dll`

- Open your Unity project
- Import the new dearVR UNITY asset into your project.
- Close Unity.

When upgrading from earlier versions than v.1.5.0, the old managed plugin components have to be replaced with the new C# source components. The dearVR manager components will loose their values and are not legacy compatible. To keep the values you used in your dearVR sources, you can update the fileID and GUID to replace the components without loosing any information. You might want to create a backup of your scenes, as the instructions are performed on the Unity scene files.

- Create backups of your Unity scenes
- Open your Unity scenes in a text editor.
- Find all
  `fileID: -1933281608, guid: 6fc5cdeb2b2664cd69b743656e623709`
  and replace them with
  `fileID: 11500000, guid: 497c625f05c190b478129f9cecc739f0`

- Find all
  `fileID: -1461815862, guid: 6fc5cdeb2b2664cd69b743656e623709`
  and replace them with
  `fileID: 11500000, guid: b51e54047889f1c4dac64a27f85a09c2`

To make this part of the process easier for you, you can also run a terminal command. You can find the commands in the quickstart guide in the dearVR UNITY asset.

- Open terminal
- Move to the folder, where your Unity scenes are
- For each scene, execute the commands stated in the quickstart guide inside the asset, where you replace scene_to_migrate with the name of your scene:

Afterwards your project is migrated to work with dearVR UNITY v1.5.1!

7.2 Upgrading on Windows

- Delete the following files from your Unity project:
    - `/Assets/dearVR/Components/dearVR_Components.dll`
    - `/Assets/dearVR/Components/dearVR_Components.xml`
    - `/Assets/dearVR/Components/Editor/dearVR_Editor.dll`

- Close Unity.
- On a 64bit Windows system delete the file
  `/Assets/Plugins/x86_64/AudioPluginDearVR.dll`

  on a 32bit Windows system delete the file
  `/Assets/Plugins/x86/AudioPluginDearVR.dll.`

- Open your Unity project
- Import the new dearVR UNITY asset into your project.
- Close Unity

When upgrading from earlier versions than v.1.5.0, the old managed plugin components have to be replaced with the new C# source components. The dearVR manager components will loose their values and are not legacy compatible. To keep the values you used in your dearVR sources, you can update the fileID and GUID to replace the components without loosing any information. The following instructions are performed on the unity scene files, so you might want to create a backup of your scenes.

- Open your Unity scenes in a text editor.
- Find all
  `fileID: -1933281608, guid: 6fc5cdeb2b2664cd69b743656e623709`
  and replace them with
  `fileID: 11500000, guid: 497c625f05c190b478129f9cecc739f0`

- Find all
  `fileID: -1461815862, guid: 6fc5cdeb2b2664cd69b743656e623709`
  and replace them with
  `fileID: 11500000, guid: b51e54047889f1c4dac64a27f85a09c2`

To make this part of the process easier for you, you can also run a terminal command. You can find the commands in the quickstart guide in the dearVR UNITY asset.

- Open cmd
- Move to the folder, where your Unity scenes are
- For each scene, execute the commands stated in the quickstart guide inside the asset, where you replace scene_to_migrate with the name of your scene:

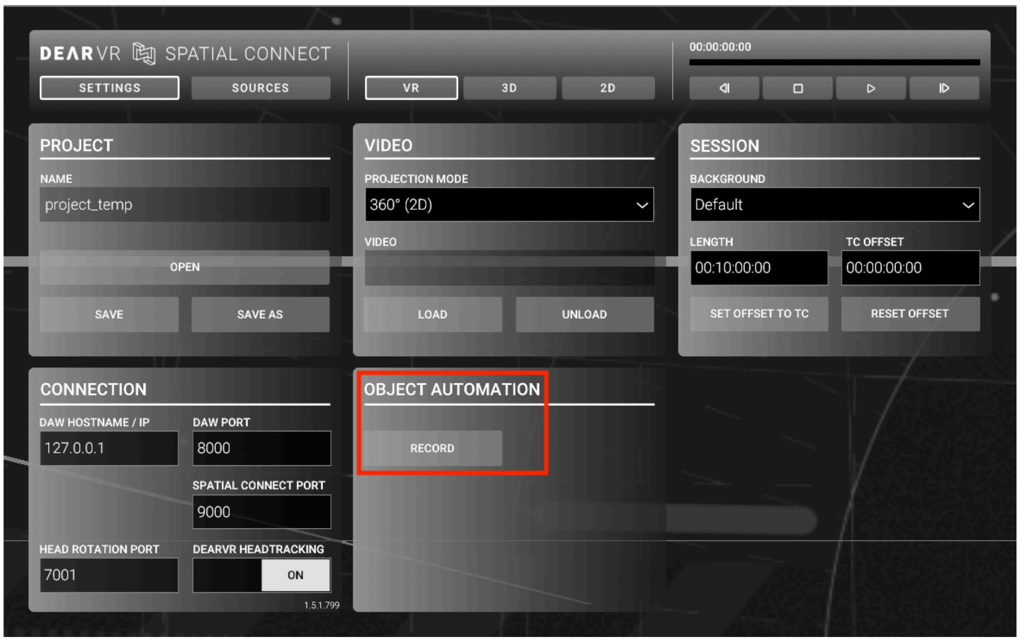
Afterwards your project is migrated to work with dearVR UNITY v1.5.1!

8 Importing automations from dearVR SPATIAL CONNECT

Premixing a sound design in a DAW can be very essential for game development. But creating the same automations in Unity per hand again should not be necessary. Therefore, we created a workflow to export these automations from SPATIAL CONNECT to use them with dearVR UNITY to get the same spatialization.

8.1 Export object automations using SPATIAL CONNECT

To export automations from SPATIAL CONNECT we added a record button to the GUI. Pressing the recording button starts the playback in the DAW and records the positions of all 3D audio sources over time. The object automations are stored in the same folder where the configuration is stored and have ".dear" as a suffix. Make sure, that you save your configuration in a place where you can find the object automations easily after recording and that you start the recording at the beginning of your audio regions to have them synchronized when importing to dearVR UNITY. The object automations naming is handled like this: spatialconnect-project-name_channel-number_channel-name.dear, where spatialconnect-project-name is the name of your spatial connect project, channel number is a counter for all channels (starting with 0) and channel-name is the name of the channel in your DAW.



PROJECT
NAME: project_temp

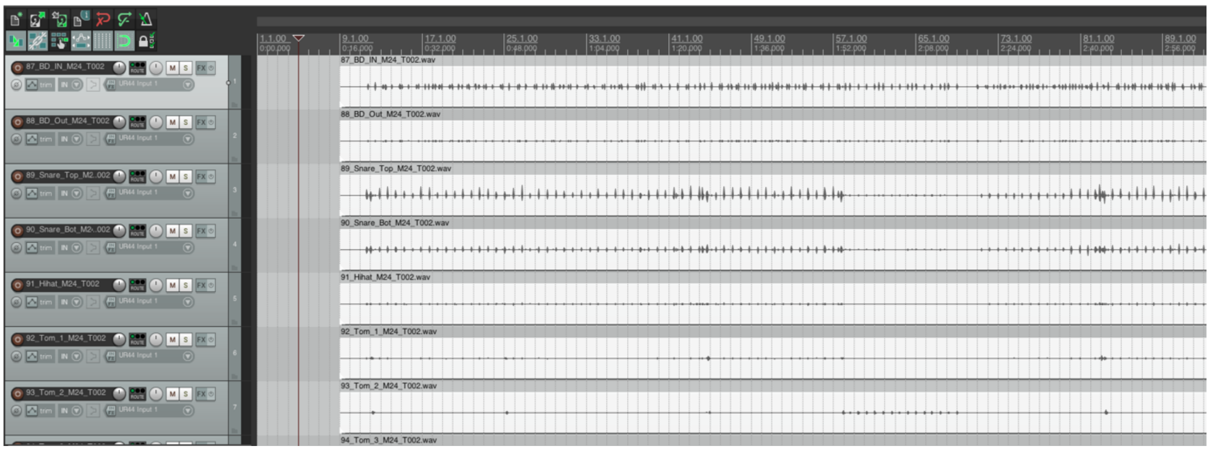
VIDEO
PROJECTION MODE: 360° (2D)
VIDEO:

SESSION
BACKGROUND: Default
LENGTH: 00:10:00:00
TC OFFSET: 00:00:00:00

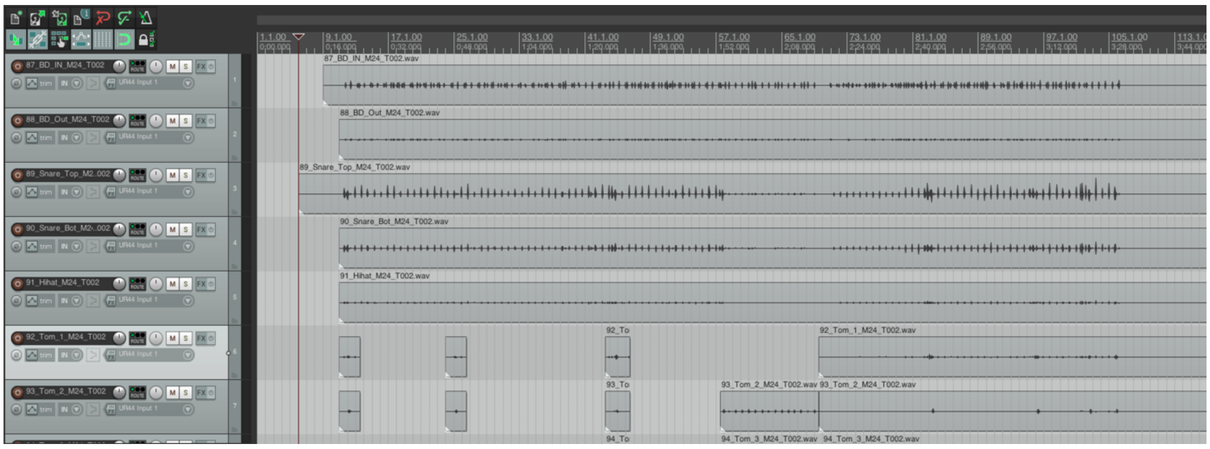
CONNECTION
DAW HOSTNAME / IP: 127.0.0.1
DAW PORT: 8000
SPATIAL CONNECT PORT: 9000
HEAD ROTATION PORT: 7001
DEARVR HEADTRACKING: ON

SPATIAL CONNECT will record all object automations you did with dearVR PRO, also the ones that don't have any explicit automation curves. Then only the position will be saved in the automation. To get the automation curve aligned with your audio, all tracks need to start at the same time and your cursor needs to be at the beginning of all audio when starting the recording of the object automations.

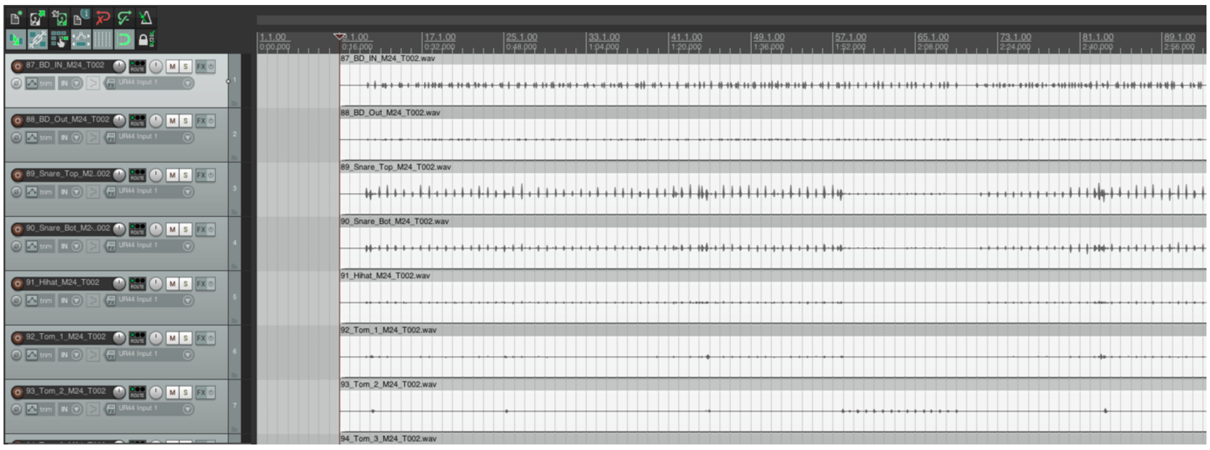
Example 1: Be careful!



Example 2: Be careful!



Example 3: This works correctly



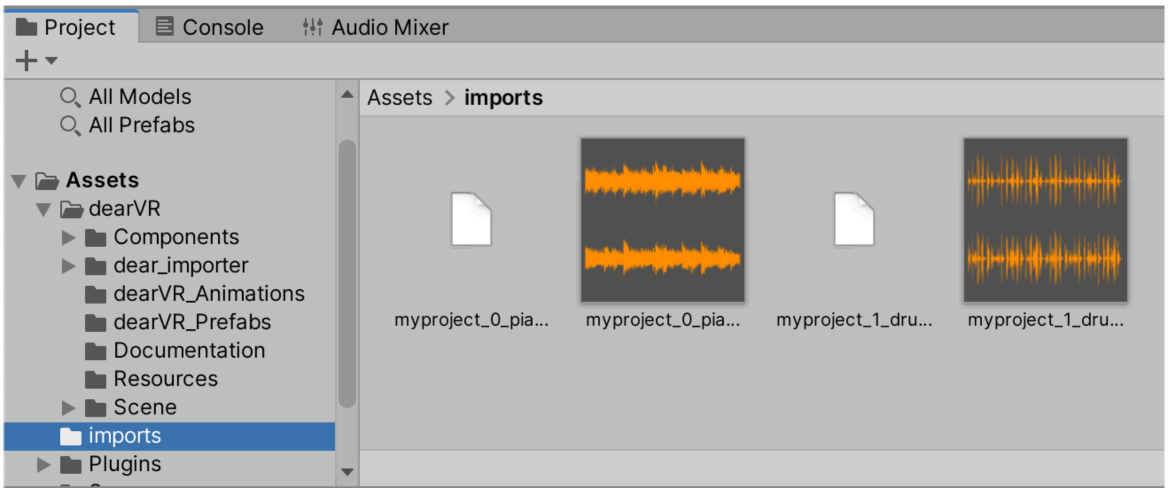
If your audio does not start all at the same time you can export the automations from a fixed point (e.g. 00:00:00:00) and then do a multichannel bounce also beginning at the same time. Many DAWs offer an option for multichannel export. Name the exported files like the created .dear files, which is “SPATIALCONNECT project name”_”channel number”_”channel name”. This helps the automations to link automatically to the audio during import.

With dearVR UNITY we deliver also an importer for the object automation files. When pulling .dear-files into the UNITY assets folder the importer automatically recognizes them as object automations and will create an automation clip and a prefab with a placeholder model for you to use in your scene. The importer also links a soundfile to the Unity audio source component if there is a clip present in the folder where you imported the object automation which has the same name as the object automation! So if you have a object automation with the name project_0_bass-drum.dear your sound file needs to have the name project_0_bass-drum.wav (or .mp3, .aac, etc.).

8.2 Importing object automations in Unity

The files created by the importer are in the following folders:

- Animation clips: `dearVR/dearVR_Animations`
- Prefabs: `dearVR/dearVR_prefabs`



The animation clips can be pulled onto game objects to link the object automation to the object. An automator component will be created automatically.

The prefabs are game objects that are already fully functional as spatialization objects. They consist of a game object that already has an animation component, a dearVR source component and an audio source component. If there was a sound file with the same name as the object automation in the folder during import the sound file is already assigned to the audio source.

All in all, the fastest way to set up your Spatial Connect project inside Unity you need to follow these steps:

- Export the object automations using Spatial Connect.
- Import the sound files you used in your project into Unity (make sure they have the same name as the object automations
- Import your object automations into Unity
- Create a new scene (or open an existing one)
- Import the prefabs into your scene
- Press play!

9 dearVR API

A complete API is available at the end of this document. To control the dearVR properties via a script, see the following `DearVRScriptLoad.cs` example:

```
using UnityEngine;
using System.Collections;
using UnityEngine.Audio;
using DearVR;

public class DearVRScriptLoad : MonoBehaviour {

    [SerializeField] bool internalReverb = false;

    AudioSource myAudioSource;

    DearVRSource myDearVRSource;

    // Assign in Inspector or in script
    [SerializeField] AudioClip myAudioClip;

    [SerializeField] DearVRSource.RoomList roomSelection;

    [SerializeField] DearVRSerializedReverb[] reverbSendList;

    [SerializeField] AudioMixerGroup audioMixer;

    [SerializeField] bool performanceMode = false;

    [SerializeField] bool loop = false;

    [SerializeField] AudioClip clipForOneShot;

    void Awake () {

        // Create dearVR-Instance
        myDearVRSource = gameObject.AddComponent<DearVRSource>();

        myDearVRSource.PerformanceMode = performanceMode;

        // Assign and set AudioSource
        myAudioSource = myDearVRSource.currentAudioSource;

        myAudioSource.loop = loop;

        if (performanceMode) {
            myAudioSource.playOnAwake = false;
        }
        // Select Room Preset
        myDearVRSource.RoomPreset = roomSelection;
        // set audiomixer
        myAudioSource.outputAudioMixerGroup = audioMixer;

        myDearVRSource.InternalReverb = internalReverb;

        if (!internalReverb) {
            if (reverbSendList != null && reverbSendList.GetLength(0) > 0) {

                myDearVRSource.SetReverbSends(reverbSendList);

            }
        }

        // Set dearVR-Settings
        myDearVRSource.BassBoost = false;


        // Assign AudioClip
        if (myAudioClip) {

            myAudioSource.clip = myAudioClip;

        } else {

            Debug.LogWarning("DEARVR: AudioClip not assigned!");

        }
    }


    public void PlayStop(bool shouldPlay) {
        if (gameObject.activeSelf) {
            if (shouldPlay) {
                DearVRPlay ();
            } else {
                DearVRStop ();
            }
        }
    }

    void DearVRPlay() {
        if (performanceMode) {
            myDearVRSource.DearVRPlay();

        } else {
            myAudioSource.Play ();

        }
    }

    public void DearVRPlayOneShot() {
        if (performanceMode) {
            myDearVRSource.currentAudioSource.loop = false;
            myDearVRSource.DearVRPlayOneShot(clipForOneShot);

        }
    }

    void DearVRStop() {
        if (performanceMode) {
            myDearVRSource.DearVRStop();

        } else {
            myAudioSource.Stop ();

        }
    }

    public void Deactivate()
    {
        gameObject.SetActive(false);
    }

    public void Activate()
    {
        gameObject.SetActive(true);
    }


    public void PlayScript()
    {
        if (myAudioSource)
        {
            myAudioSource.Play();
        }
    }
```

```
    void Update() {
        if (myDearVRSource.RoomPreset != roomSelection) {
            myDearVRSource.RoomPreset = roomSelection;
        }

        if (myDearVRSource.InternalReverb && !internalReverb) {
            myDearVRSource.InternalReverb = false;


            if (reverbSendList != null && reverbSendList.GetLength(0) > 0) {

                myDearVRSource.SetReverbSends(reverbSendList);

            }
        }
        if (!myDearVRSource.InternalReverb && internalReverb) {
            myDearVRSource.InternalReverb = true;
        }
    }
}
```

10 Troubleshooting

1. Audible dropouts in consequence of performance issues.
Please check the DSP load in Stats Window. Reduce the performance load by using Reverb Sends.

2. Heavy DSP-Processing load while only playing a few spatial audio sources.
Check if you have further (not playing) audio sources in the scene. Activate the Performance Mode for each.

2. Performance Issues on Android Devices.
For android devices, always set DSP Buffer Size in the Audio Manager to Best Performance.

3. Problems with playback speed or audio stuttering on mobile devices.
Audio stuttering or slow playback speed might occur as a result of performance issues. Please be sure to follow the advice given to performance issues.

4. Function `AudioMixerSnapshot.TransitionTo`
Transitions between snapshots are a great feature within Unity. However, it is not possible to change the parameter for Room Presets and Reverb IDs when using the function `TransitionTo`.

5. Distant sound source are still audible in reverb
Check Settings for Distance Attenuation. Using the unity distance attenuation (see 0 – Settings), the Reverb Zone graph is responsible for the Reverb Level over distance (see Illustration 4.8 - Audio Source Graph).

6. XCode Bitcode
XCode Bitcode option is not supported yet**. Disable Bitcode** in Build Settings / Build Options.

7. External Distance Attenuation
The External Distance Attenuation is not available in Unity 5.6 and 2017.1.

8. Legacy issues

Updating from version 1.2.1 or earlier to v1.5.1 causes the loss of the dearVR Manager's settings.

Contact us to report bugs, errors or suggest features via support@dear-reality.com

Please include the following information:

- Plugin version
- Operating system
- Logfile / Console output
- Host software

11 Changelog

dearVR UNITY v1.6

- Add object automation workflow
- Fixed minor bugs

dearVR UNITY v1.5.1

- Add support for Android ARM64
- Fixed minor bugs

dearVR UNITY v1.5.0

- Add support for IL2CPP
- Up to 60% performance boost on binaural output
- Access to C# editor scripts
- Optimized dearVR API documentation
- Various bug fixes

dearVR UNITY v1.2.1

- add API documentation
- update dearVR demo scene
- update *DearVRScriptLoad.cs* example
- disabled Bitcode for iOS
- Fixed minor bugs

dearVR UNITY v1.2.0

- Added Obstruction feature
- Optimized Performance for Internal Reverb
- Optimized Auralization
- Fixed minor bugs

dearVR UNITY v1.1.1

- Fixed input channel parameter to 0.0
- Fixed not playing sources after deactivation and reactivation

- Enabled mono audio clips for spatialization
- Enabled ENABLE_BITCODE for iOS
- Enabled DearVRPlayOnAwake on OnEnable
- Auto detecting Android architecture as ARMv7 now

dearVR UNITY v1.1

- Changed Distance Correction range to [0.01 m - 10 m]
- Changed Occlusion default to 0.6
- Changed Occlusion Update Interval default to 0.2
- Changed Occlusion ray cast start to Audio Listener (was Camera before)
- Added Force Occlusion flag
- Added Mute (no processing) flag to Reverb Bus
- Added Performance-Mode: not processing spatial audio sources, that aren't playing.
- Added Reverb Tail after stop in Performance-Mode (only useful for Internal Reverb)
- Solved bug in Windows on performance overload

12 Contact

Support

Please let us know if there are any questions concerning the dearVR Plugin.
If you need further assistance, please send an email to:
support@dear-reality.com

For the latest news concerning dearVR, please visit our website at:
www.dearVR.com

Dear Reality GmbH
Binterimstraße 8
40223 Düsseldorf

Caution

Using headphones requires responsible listening. Damage to hearing occurs when listening to loud sounds with headphones over time.

- Set the volume control of your computer to a minimum when connecting your headphones.
- Set the volume in a quiet environment and select the lowest volume at which you can hear adequately.
- Do not turn the volume control too high, as this can cause permanent hearing damage.
- Be aware that you can adapt to higher volume settings over time, not realizing that the higher volume may be harmful to your hearing.

Dear Reality GmbH will, in any event, not be liable for any damage to hearing caused by loud sounds.

Class DearVRSource

Dear VR source, equivalent to Unity's AudioSource, is responsible for object based binaural sounds This is the class you would need to put on your objects, almost all aspects of sound can be adjusted from this class.

Inheritance

↳ System.Object
  ↳ DearVRSource

Namespace: DearVR (DearVR.html)

Assembly: cs.temp.dll.dll

Syntax

```
public class DearVRSource : MonoBehaviour
```

Fields

clipIsPlaying

The clip on the source is currenlty playing.

Declaration

```
public bool clipIsPlaying
```

Field Value

Type	Description
System.Boolean	

obstructionRayUpdateTime

how often should ray casting be performed for obstruction.

Declaration

```
public float obstructionRayUpdateTime
```

Field Value

Type	Description
System.Single	

obstructionWallMask

The obstruction mask. Used for ray tracing obstructing objects. all objects having this layer, will be considered for obstruction

Declaration

```
public LayerMask obstructionWallMask
```

Field Value

Type	Description
LayerMask	

occlusionRayUpdateTime

how often should ray casting be performed for occlusion.

Declaration

```
public float occlusionRayUpdateTime
```

Field Value

Type	Description
System.Single	

occlusionWallMask

The occlusion mask. Used for ray tracing occluding objects. all objects having this layer, will be considered for occlussion

Declaration

```
public LayerMask occlusionWallMask
```

Field Value

Type	Description
LayerMask	

Properties

Auralization

turns realtime auralization. This uses realtime room geometries to calculate realtime reflections. NOTE: only works if either RoomAnalyzer (DearVR.DearVRManager.html#DearVR_DearVRManager_RoomAnalyzer) or SetRoomGeo (DearVR.DearVRManager.html#DearVR_DearVRManager_SetRoomGeo) are set to true.

Declaration

```
public bool Auralization { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if auralization; otherwise, `false`.

AzimuthCorrection

Gets or sets the azimuth correction. Can be used to scale the positioning of sounds

Declaration

```
public float AzimuthCorrection { get; set; }
```

Property Value

Type	Description
System.Single	The azimuth correction.

BassBoost

filter, used to boost up the low end. Sometimes this is necessary to build up more low end, when binauralising sounds.

Declaration

```
public bool BassBoost { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if bass boost; otherwise, `false` .

Bypass

if set to true bypasses the dearVR engine.

Declaration

```
public bool Bypass { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if bypass; otherwise, `false` .

currentAudioSource

Gets the current audio source, being processed by dearVR

Declaration

```
public AudioSource currentAudioSource { get; }
```

Property Value

Type	Description
AudioSource	The current audio source.

DearVRPlayOnAwake

should the audio play on awake

Declaration

```
public bool DearVRPlayOnAwake { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if dear VR play on awake; otherwise, `false` .

DirectLevel

Direct gain of the source, before reverb and reflection.

Declaration

```
public float DirectLevel { get; set; }
```

Property Value

Type	Description
System.Single	The direct level.

DistanceCorrection

Gets or sets the distance correction. Can be used to scale the positioning of sounds

Declaration

```
public float DistanceCorrection { get; set; }
```

Property Value

Type	Description
System.Single	The distance correction.

ForceObstruction

obstruction should be active for this source, irrespective of ray casting.

Declaration

```
public bool ForceObstruction { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if force obstruction; otherwise, `false` .

ForceOcclusion

occlusion should be active irrespective of ray casting.

Declaration

```
public bool ForceOcclusion { get; set; }
```

Property Value

Type	Description

Type	Description
System.Boolean	`true` if force occlusion; otherwise, `false` .

GainLevel

master gain of audio.

Declaration

```
public float GainLevel { get; set; }
```

Property Value

Type	Description
System.Single	The gain level.

InputChannel

which side of stereo channel to use for binaural processing. NOTE: Due to an update in Unity's Spatial framework, changing this would not have any effect. Even with stereo files, only the left channel is used as input

Declaration

```
public float InputChannel { get; set; }
```

Property Value

Type	Description
System.Single	The input channel.

InternalReverb

if set to true, this instance uses its own reverb engine, independent of the reverb send plugin

Declaration

```
public bool InternalReverb { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if internal reverb; otherwise, `false` .

IsProcessing

Gets a value indicating whether the engine is actually processing audio.

Declaration

```
public bool IsProcessing { get; }
```

Property Value

Type	Description

Type	Description
System.Boolean	`true` if this instance is processing; otherwise, `false` .

ObstructionActive

activates obstruction for this source

Declaration

```
public bool ObstructionActive { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if obstruction activ; otherwise, `false` .

ObstructionDebugRayCast

Show debug ray casts used for obstruction

Declaration

```
public bool ObstructionDebugRayCast { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if occlusion debug ray cast; otherwise, `false` .

ObstructionLevel

how much obstruction should have an effect on the sound

Declaration

```
public float ObstructionLevel { get; set; }
```

Property Value

Type	Description
System.Single	The obstruction level.

ObstructionRayUpdateFreq

ray casting frequency used for obstruction detection

Declaration

```
public float ObstructionRayUpdateFreq { get; set; }
```

Property Value

Type	Description
System.Single	The obstruction ray update freq.

OcclusionActive

activates occlusion for this source

Declaration

```
public bool OcclusionActive { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if occlusion activ; otherwise, `false` .

OcclusionDebugRayCast

Show debug ray casts used for occlusion

Declaration

```
public bool OcclusionDebugRayCast { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if occlusion debug ray cast; otherwise, `false` .

OcclusionLevel

How much occlusion should have and effect on the sound

Declaration

```
public float OcclusionLevel { get; set; }
```

Property Value

Type	Description
System.Single	The occlusion level.

OcclusionRayUpdateFreq

ray casting frequency used for occlusion detection

Declaration

```
public float OcclusionRayUpdateFreq { get; set; }
```

Property Value

Type	Description
System.Single	The occlusion ray update freq.

PerformanceMode

activating this bypasses processing during idle state. Play Audio Sources only with DearVRPlay() (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlay) , DearVRPlayOneShot(AudioClip) (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlayOneShot_AudioClip_) or DearVRPlayOn Awake (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlayOnAwake) flag. WARNING: Do not use the Play() or PlayOnAwake flag in Performace Mode!"

Declaration

```
public bool PerformanceMode { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if performance mode; otherwise, `false` .

ReflectionLevel

Gets or sets the reflection level.

Declaration

```
public float ReflectionLevel { get; set; }
```

Property Value

Type	Description
System.Single	The reflection level.

ReflectionLP

lowpass filter frequency on reflections

Declaration

```
public float ReflectionLP { get; set; }
```

Property Value

Type	Description
System.Single	The reflection L.

ReverbLevel

Gets or sets the reverb level.

Declaration

```
public float ReverbLevel { get; set; }
```

Property Value

Type	Description
System.Single	The reverb level.

ReverbLP

lowpass filter frequency on reverb

Declaration

```
public float ReverbLP { get; set; }
```

Property Value

Type	Description
System.Single	The reverb L.

ReverbStopOverlap

Gets or sets the reverb tail in seconds, if using DearVRPlay() (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlay) or DearVRPlayOneShot(AudioClip) (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlayOneShot_AudioClip_), this variable is used to deactivate processing in engine

Declaration

```
public float ReverbStopOverlap { get; set; }
```

Property Value

Type	Description
System.Single	The reverb stop overlap.

RoomPreset

Gets or sets the room preset.(aka virtual acoustic preset)

Declaration

```
public DearVRSource.RoomList RoomPreset { get; set; }
```

Property Value

Type	Description
DearVRSource.RoomList (DearVR.DearVRSource.RoomList.html)	The room preset.

RoomSize

Sets the roomsize, that can be used to adjust the reverb tail. If Auralization (DearVR.DearVRSource.html#DearVR_DearVRSource_Auralization) is on, then this also has an effect on reflections

Declaration

```
public float RoomSize { get; set; }
```

Property Value

Type	Description
System.Single	The size of the room.

UseUnityDistance

should the engnine use unity distancec attenaution?.

Declaration

```
public bool UseUnityDistance { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if use unity distance; otherwise, `false` .

Methods

DearVRPlay()

plays the source in performance mode, Unity processes all audio sources irrespective of them playing or not, this method circumvents that problem, by deactivating the processing in dearVR if you use this method you have to use DearVRStop() (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRStop) since this accounts for the reverb tail before stopping dearVR processing, Also you can change ReverbStopOverlap (DearVR.DearVRSource.html#DearVR_DearVRSource_ReverbStopOverlap) to account for different reverb tails after which the actual stop takes place.

Declaration

```
public void DearVRPlay()
```

DearVRPlayOneShot(AudioClip)

plays the source in performance mode and with oneshot, essentially calling unity's PlayOneShot UnityEngine.AudioSource.PlayOneShot DearVRPlay() (DearVR.DearVRSource.html#DearVR_DearVRSource_DearVRPlay)

Declaration

```
public void DearVRPlayOneShot(AudioClip clip)
```

Parameters

Type	Name	Description
AudioClip	*clip*	Clip.

DearVRStop()

Stops the audio and then processing after ReverbStopOverlap (DearVR.DearVRSource.html#DearVR_DearVRSource_ReverbStopOverlap) (in seconds) is over

Declaration

```
public void DearVRStop()
```

GetReverbSendList()

returns an instance of current reverb send structure, can be used to edit the reverb structure with a subsequent call to SetReverbSends(DearVRSerializedReverb[]) (DearVR.DearVRSource.html#DearVR_DearVRSource_SetReverbSends_DearVR_DearVRSerializedReverb___)

Declaration

```
public DearVRSerializedReverb[] GetReverbSendList()
```

Returns

Type	**Description**
DearVRSerializedReverb (DearVR.DearVRSerializedReverb.html)[]	The reverb send list.

ReverbSendsChanged()

used internaly by dearVR, calling this function would not have any effect from client side

Declaration

```
public bool ReverbSendsChanged()
```

Returns

Type	**Description**
System.Boolean	`true`, if sends changed was reverbed, `false` otherwise.

SetReverbSends()

used internaly by dearVR, do not use this function to set reverb sends instead use GetReverbSendList() (DearVR.DearVRSource.html#DearVR_DearVRSource_GetReverbSendList) and SetReverbSends(Dear VRSerializedReverb[]) (DearVR.DearVRSource.html#DearVR_DearVRSource_SetReverbSends_DearVR_DearVRSerializedReverb___)

Declaration

```
public void SetReverbSends()
```

SetReverbSends(DearVRSerializedReverb[])

use this fuction to set the the reverb sends, these will be used to send the binauralised sound to the reverb bus only used if internal reverb is inactive

Declaration

```
public void SetReverbSends(DearVRSerializedReverb[] rl)
```

Parameters

Type	**Name**	**Description**
DearVRSerializedReverb (DearVR.DearVRSerializedReverb.html)[]	*rl*	Rl.

Class DearVRManager

Dear VR manager. A singleton manager class for dearVR global settings

Inheritance

↳ System.Object
  ↳ DearVRManager

Namespace: DearVR (DearVR.html)

Assembly: cs.temp.dll.dll

Syntax

```
public class DearVRManager : MonoBehaviour
```

Properties

Bypass3DAudio

DearVRManager (DearVR.DearVRManager.html) Bypass the bniaural rendering for all sources, reflections and reverb processing are still active. effectively enabling speaker mode. Note: Loudspeaker Mode enables you to switch from 3D Audio for headphones to a loudspeaker compatible mix. Distance attenuation, Reflection, and Reverb levels are still maintained.

Declaration

```
public static bool Bypass3DAudio { get; set; }
```

Property Value

Type	**Description**
System.Boolean	`true` if bypass3 D audio; otherwise, `false` .

DearListener

Gets or sets the listener used by dearVR.

Declaration

```
public static AudioListener DearListener { get; set; }
```

Property Value

Type	**Description**
AudioListener	The dear listener.

DebugRoomAnalyzer

turns room analyzer debuging rays off/on

Declaration

```
public static bool DebugRoomAnalyzer { get; set; }
```

Property Value

Type	Description
System.Boolean	

FrontBackGeo

Gets or sets the front side and back side of room geometry.

Declaration

```
public static Vector2 FrontBackGeo { get; set; }
```

Property Value

Type	Description
Vector2	The front back geo.

Instance

Use this to access the singleton instance of DearVRManager

Declaration

```
public static DearVRManager Instance { get; }
```

Property Value

Type	Description
DearVRManager (DearVR.DearVRManager.html)	The instance.

LeftRightGeo

Gets or sets the left side and the right side of room geometry.

Declaration

```
public static Vector2 LeftRightGeo { get; set; }
```

Property Value

Type	Description
Vector2	The left right geo.

RoomAnalyzer

if set to true, room geometry is calculated by ray tracing and sent to the engine and used for reflections turning this on would turn SetRoomGeo (DearVR.DearVRManager.html#DearVR_DearVRManager_SetRoomGeo) off

Declaration

```
public static bool RoomAnalyzer { get; set; }
```

Property Value

Type	Description

Type	Description
System.Boolean	`true` if set room geo; otherwise, `false`.

RoomMask

layers used for the raytracing of room geoemtries.

Declaration

```
public static LayerMask RoomMask { get; set; }
```

Property Value

Type	Description
LayerMask	The room mask.

RoomUpdateFreq

how often room geometries are updated with ray tracing

Declaration

```
public static float RoomUpdateFreq { get; set; }
```

Property Value

Type	Description
System.Single	

SetRoomGeo

if set to true, room geometry is sent to the engine and used for reflections, turning this on will turn off Room Analyzer (DearVR.DearVRManager.html#DearVR_DearVRManager_RoomAnalyzer)

Declaration

```
public static bool SetRoomGeo { get; set; }
```

Property Value

Type	Description
System.Boolean	`true` if set room geo; otherwise, `false`.

UpDownGeo

Gets or sets up side and down side of room geometry.

Declaration

```
public static Vector2 UpDownGeo { get; set; }
```

Property Value

Type	Description

Type	Description
Vector2	Up down geo.