dearVR

3D audio reality engine

USER MANUAL
v1.2.1

Please read this manual carefully before using the software.

Using headphones requires responsible listening!

Last updated: October 2017

Quick Start Guide

1. Import

- Import the ***dearVR Asset*** to your project.
  For details on how to import a *unitypackage* 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 ***DearVRManager*** component to a game object in your scene.
- Create a new game object and add the ***DearVRSource*** 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 *DearVRSource* anytime. Press keys **"R"** and **"T"** to switch presets and keys **"F"** and **"G"** to switch audioclip.

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 top* and select ***dearVR Reverb***.
   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 ***dearVRSource*** 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 signal to.

   Note: SEND determines the individual Send Level to a Reverb Plugin while **REVRB 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!

Updating dearVR within an existing project:

OSX:

Just import the new dearVR Asset and restart Unity.

WINDOWS:

1. Close Unity

2. Use the Windows Explorer to delete the file ***AudioPluginDearVR.dll*** in your projects folder ***Assets/Plugins/x86_64*** (or *x86*, depending on your system).

3. Open your Unity Project

4. Import the new dearVR Asset.

5. Restart Unity.

Note: Not following these steps leads to the warning *"Copying file failed"* because AudioPluginDearVR.dll is still loaded in the Unity Editor.

Table of Contents

1.Introduction

Thank you for purchasing our dearVR Plugin for Unity and welcome to the next step of immersive audio production. With the *dearVR audio reality engine* you are able to design deep auditory worlds and fantastic sound experience within Unity for all common platforms. This manual will help you understand the *dearVR* Plugin and how to use it in your projects.

I**mportant Note:**
dearVR is a 3D Audio technology for headphones. Any kind or brand will do, but without a headphone it won`t work properly. Please check to have your left and right ear pieces properly placed and let's get started…

Have fun!

1.1 About 3D Audio

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.

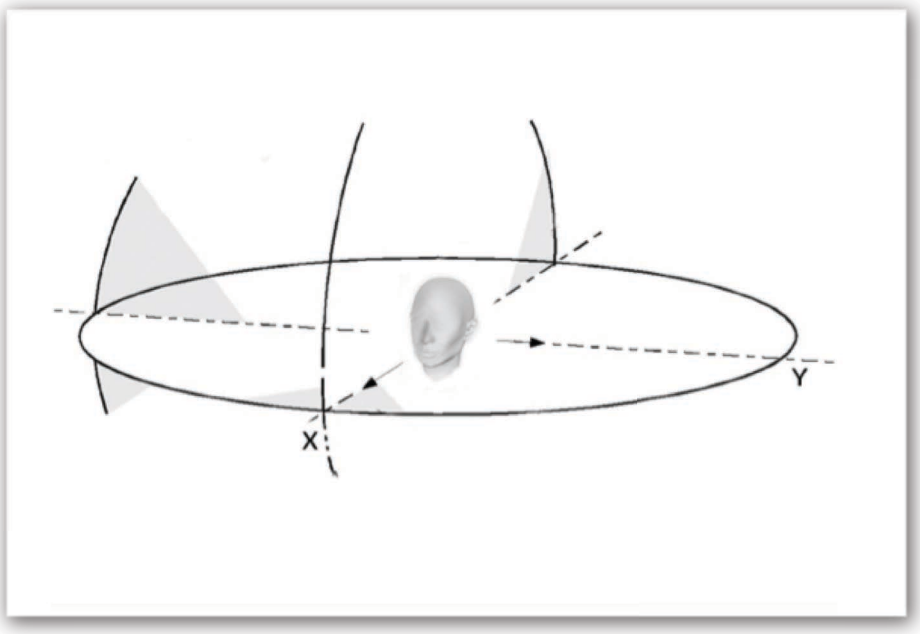

Illustration 1.1 - Full 360 degree 3D audio sphere

This gives you the ability to position a sound object all around the listener - behind, in front of, to the right or to 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 actually 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 localisation of a sound source. The head tracking technology, which is part of virtual reality devices, offers a solution to this problem.

Binaural 3D Audio is not made for listening via stereo speakers. Although in general the playback is possible, you will have to face strong colorations of the sound depending on your playback system. The reason is crosstalk, meaning a large portion of the left speaker signal will also go to the right ear of the listener. Similarly a large portion of the right speaker signal will go 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 own ears.

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

1.3 Version

This manual is for *dearVR* Version 1.2.1

2. Installation

This chapter describes how to setup 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.

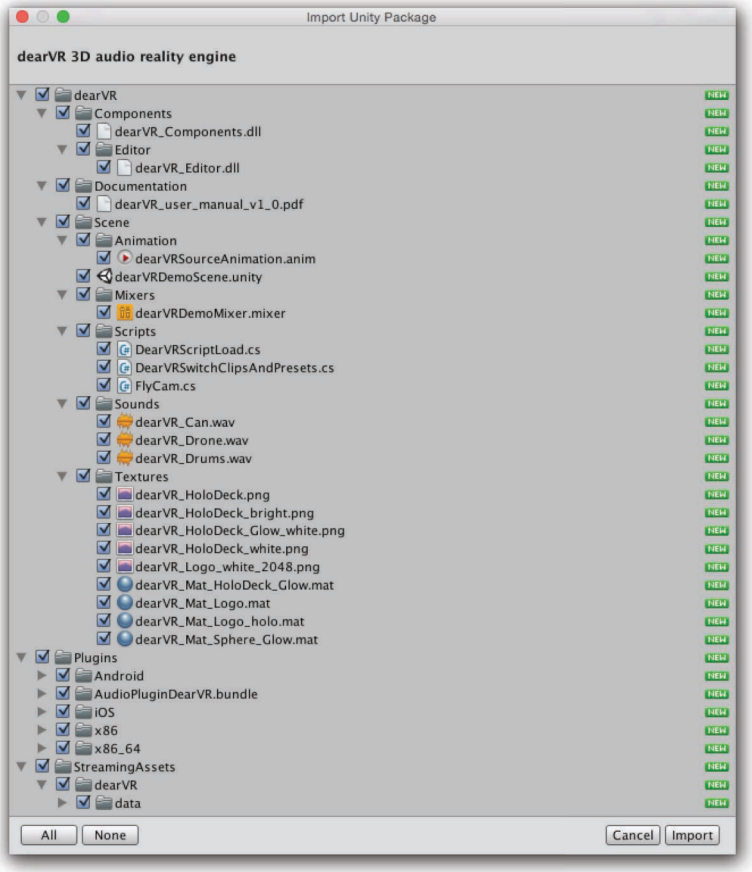

Illustration 2.2 - Import Asset

2.3 The dearVR folders

After import three new folders appear inside your project:

1. dearVR folder

2. Plugins folder

3. StreamingAssets folder

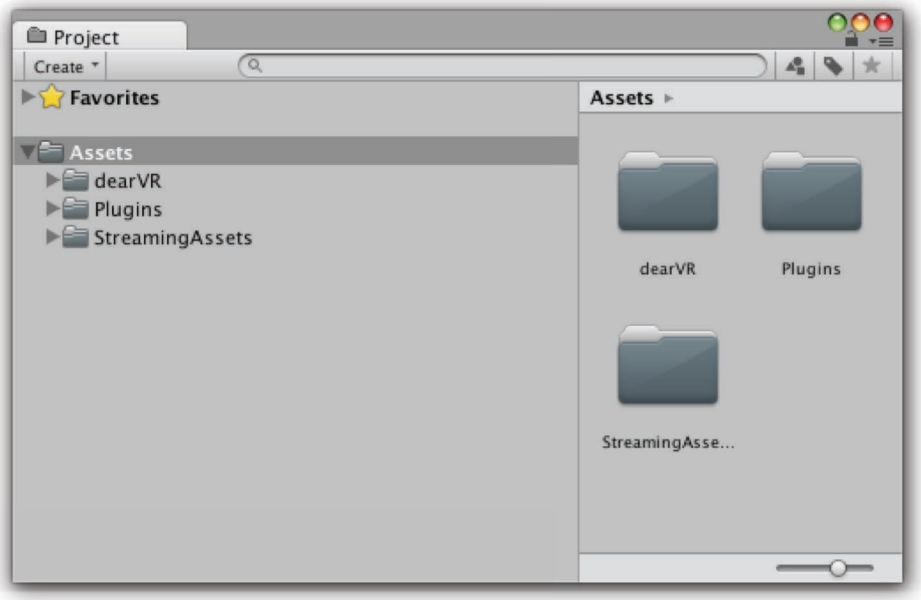

Illustration 2.3.1 - Imported folders in the Project Window

- The *dearVR* folder contains:
  a) *dearVR Demo* - demo scene to get a first impression of the Plugin.
  b) Components - *dearVR Manager* and *dearVRSource*.

  Note: *dearVREditor* is an internal system component.
- The *Plugins* folder contains the native plugins for OSX, Windows, iOS and Android.
- The *StreamingAssets* folder contains necessary data files of the dearVR Engine.

Dynamically linked libraries (DLLs)

The dearVR Asset contains DLLs instead of scripts. Usually, scripts in Unity are kept in a project as source files and compiled whenever the source changes. However, it is also possible to compile a script to a **dynamically linked library (DLL)** using an external compiler. The resulting DLL can then be added to the project and the classes it contains can be attached to objects just like normal scripts.

You can attach the dearVRSource or dearVRManager component to a game object just like a script. The dearVR_Editor component is an internal system component for the dearvR engine. Please don`t change the components platform import settings.

For more information about DLLs please refer to the Unity manual:
http://docs.unity3d.com/Manual/UsingDLL.html

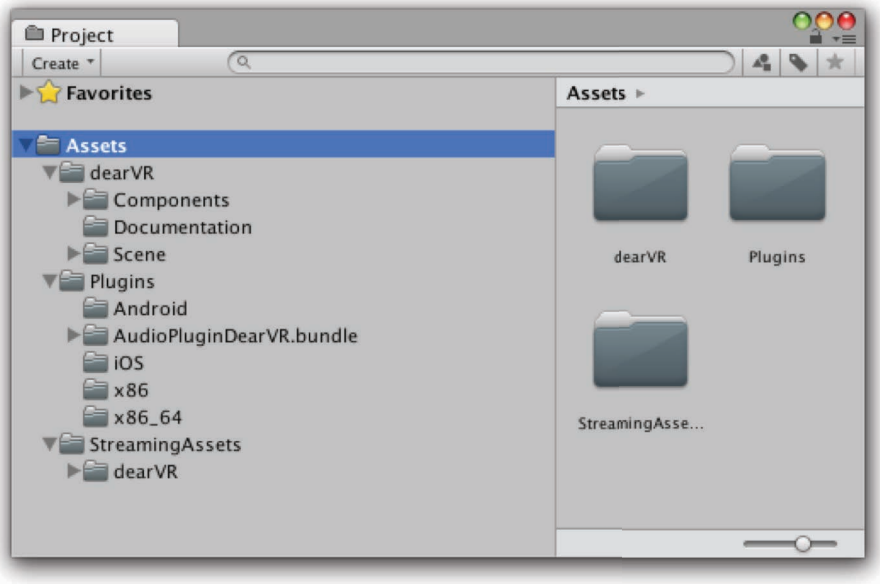

Illustration 2.3.2 – Folder content dearVR Plugin

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 will find the Spatializer Plugin Menu.
- Choose *dearVR Engine*.

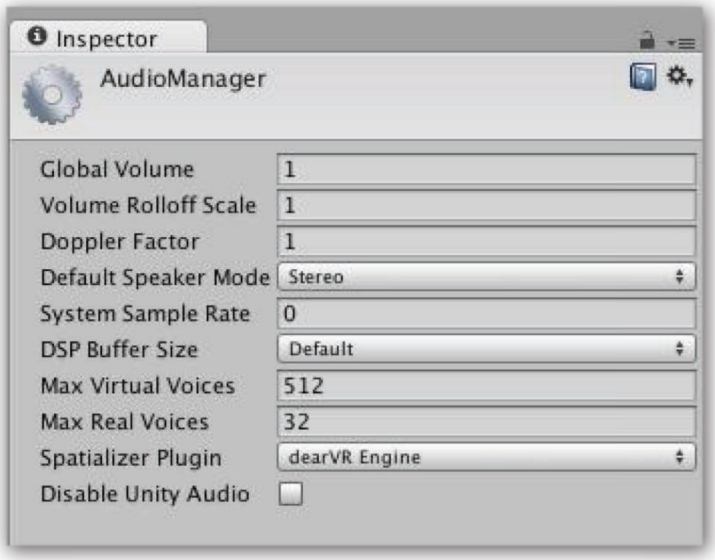

Illustration 2.4 - Unity AudioManager Spatializer Plugin

Note:

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

2.5 Getting Started

How to configure a new scene:

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

Done - that's all!

IMPORTANT:
Audioclips for dearVR Source have to be a stereo file!
Convert mono files to stereo files with a third party tool before using as a dearVR Source.

3. Overview 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 great amount of realism.

The virtualisation of an acoustic environment with object based sound sources refers to combining 3D spatial location cues with distance, motion, and ambience. 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, a mono audio track is converted 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 localisation and the impression of sounds coming from outside of the head.

Early reflections can be modelled 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 reflections properties are recalculated 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 or locations and created over 40 room presets. You can easily 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 will find inside the dearVR Asset and their corresponding parameters. Any parameter can be changed in real-time during playback.

1. *dearVRSource* 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 dearVRSource

For binaural processing each Unity AudioSource requires the *dearVRSource* component. Once you have added the component to a game object, an audio source component will be generated automatically if it does not already exist.

There are 4 main parameter sections within the inspector window of a *dearVRSource* component:

- Reverb
- Levels
- Settings
- Occlusion

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

4.1.1 Reverb

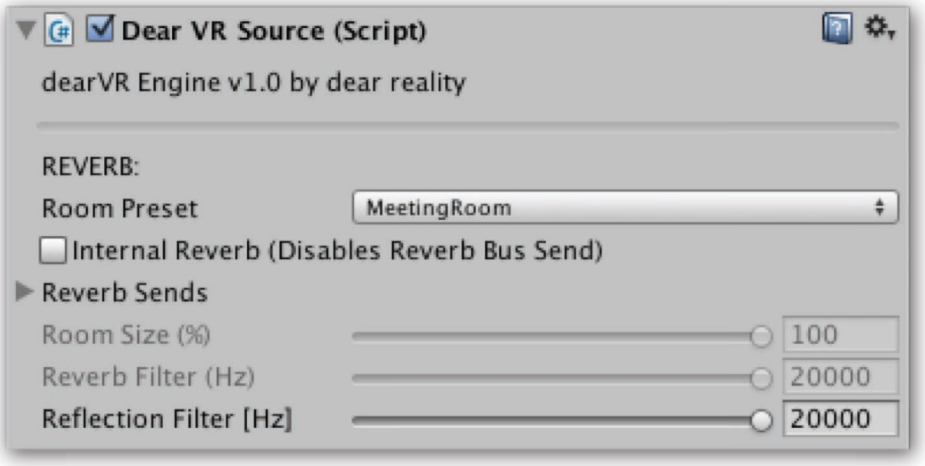

Illustration 4.1.1 – Reverb parameter

Room Preset	Menu to select different room presets. (Refer to 5.6 Preset List) **Note:** The larger 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. Default slider position is 20000 Hz for internal preset value! **Note:** Each preset has an optimized internal value for the reverb filter – used if slider is set to 20000 Hz.
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. Default slider position is 20000 Hz for internal preset value! **Note:** Each preset has an optimized internal value for the reverb filter – used if slider is set to 20000 Hz.

4.1.1.1 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.

Note: Reverb Sends enable crossfading between different reverbs – e.g. needed for sound sources moving from one room to another.

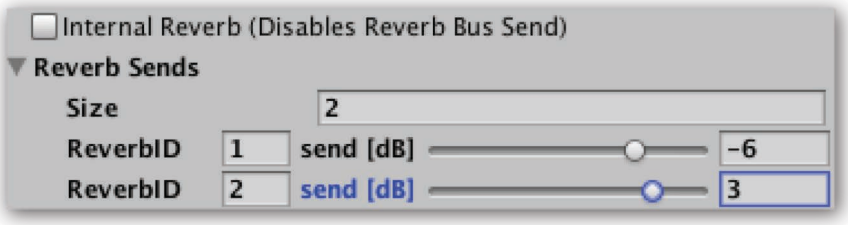

Illustration 4.1.1.1 – Reverb Sends

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. (A unique Reverb ID has to be defined 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

Illustration 4.1.2 – Levels

Gain Level	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) **Note:** Values for direct, reflection and reverb levels influence the distance perception.
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

4.1.3 Settings

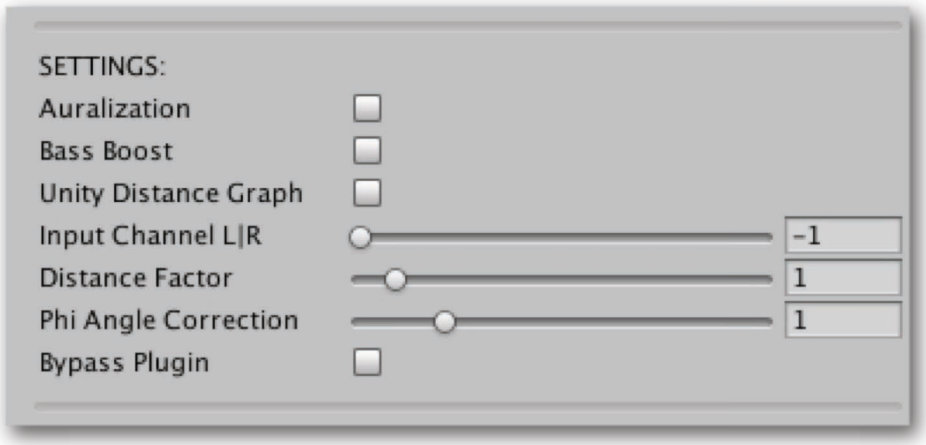

Illustration 4.1.3 – Settings

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 analysed with the real-time room analyser (for details see 4.2 dearVRManager), geometry is determined by the active room preset.
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 4.1.5.) **Default:** OFF **Note:** Maximum distance value for the internal dearVR processing is set to 28 meters. You can modify this value by using the distance correction parameter.
Input Channel	Sets dearVR input signal from the stereo audio clip. Left channel (-1), Right channel (+1) or a mix (-1 < x > +1). **Default:** (-1) Left channel **Important:** A stereo file is required for dearVR processing!
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 lead to an increased distance perception. Value < 1.0 lead to an 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 centered 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, the acoustic insulation of a wall, window or door can be simulated.

TIP: Occlusion alters direct signal and reverberation signal of an audio source.

Note: Occlusion is always measured between audio listener and audio source.

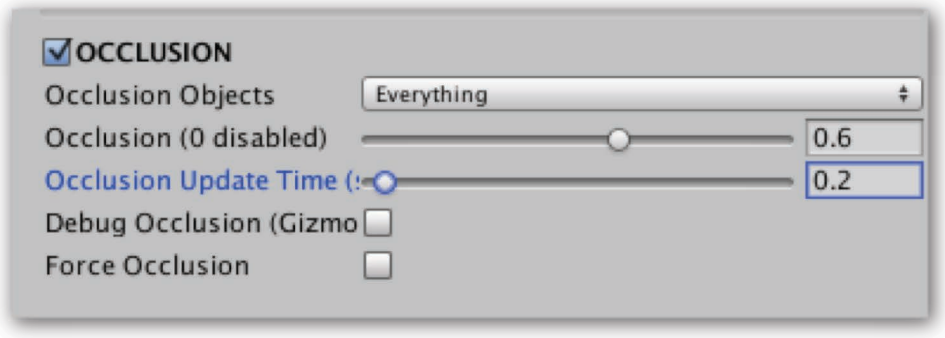

Illustration 4.1.4 – Occlusion

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 raycasting 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.

TIP: Obstruction mainly influences the direct signal of an audio source. The reverberation signal gets less influenced, if an object is obstructed.

Note: Obstruction is always measured between audio listener and audio source.

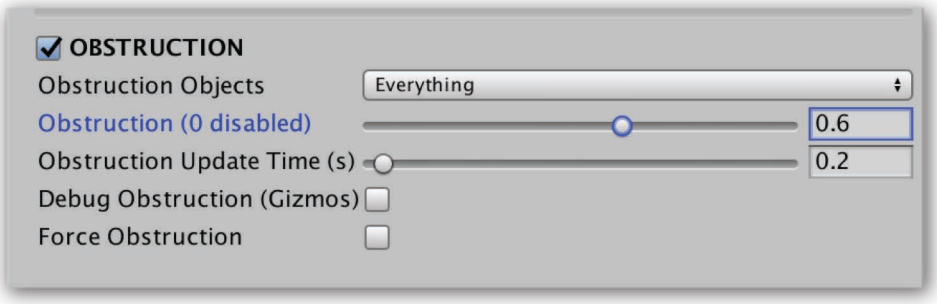

Illustration 4.1.5 – Obstruction

Obstruction Objects	Select the layer of objects that lead to obstruction.
Obstruction	Sets the level of obstruction if a sound source is obstructed by another object (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 raycasting 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 processing of spatialized audio sources if they are in idle state. Usually all audio sources with the Spatialize checkbox enabled will be 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 Performace Mode!**

Illustration 4.1.6 – Performance Mode

dearVR Play Performance Mode	Activates Performance Mode. Spatial audio sources are NOT processed during 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, 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 *dearVRSource* component is added.
Bypass Effects / Listener Effects / Reverb Zones	These Settings don't have any effect on the dearVR rendering.
Stereo Pan	No effect
Spatial Blend	Automatically set to 2D if a *dearVRSource* component is added. **Important:** Parameter has to be kept at 2D!
Reverb Zone Mix	If Unity Distance Graph in dearVRSource enabled, slider or graph control Reverb Level.
Doppler Level	No effect (will be implemented in future updates)
Spread	No effect
3D Sound Settings Graph	**Volume** graph alters the Direct Level over distance. **Reverb Zone** graph alters the Reverb Level over distance. **Note:** If distance is farther than **Max Distance** value, the farthest value is processed. Set **Volume** and **ReverbZone** to zero at **Max Distance** to mute both for higher distances.

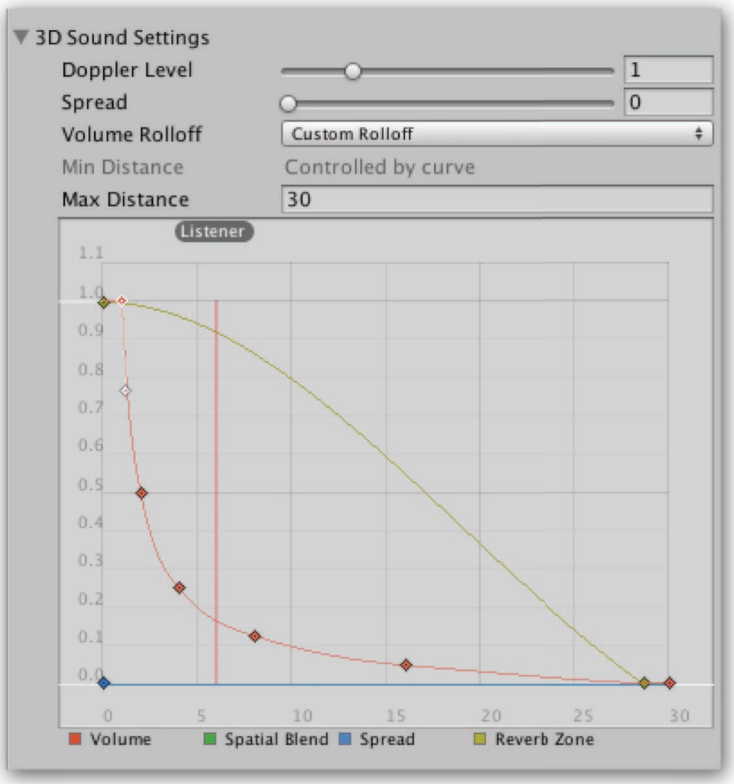

Illustration 4.1.7 – Audio Source Graph

4.2 dearVR Manager

The dearVR Manager component is mandatory in each scene or in a global root scene. It defines global setting 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 main parameters for auralization and room analyser.

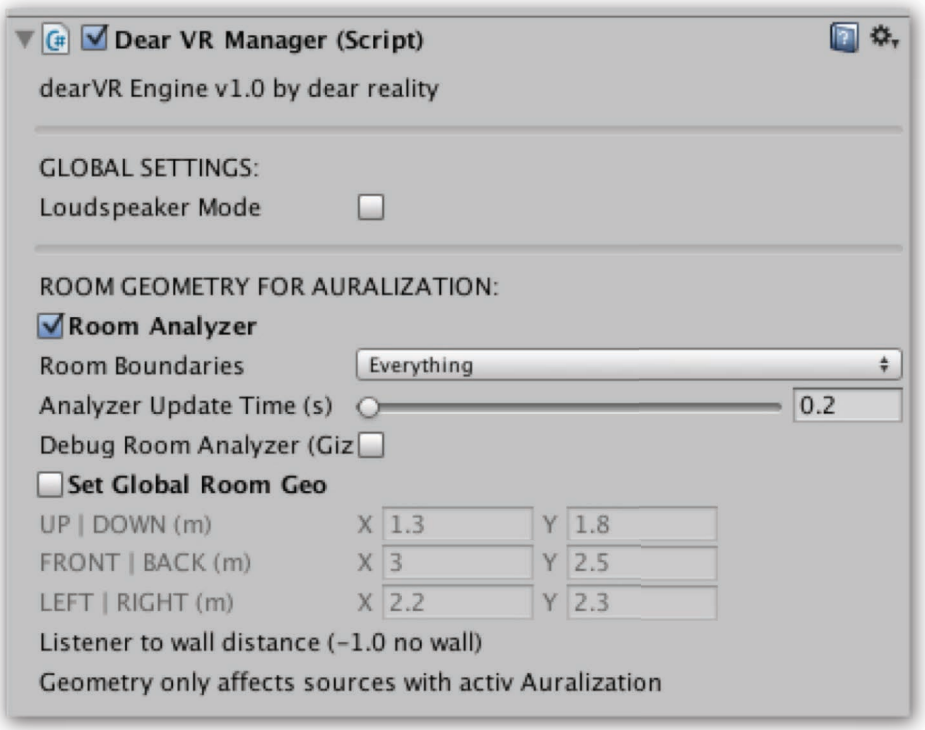

Illustration 4.2 – Dear VR Manager

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.
Room Analyzer	Enable the real-time room analyzer to get scene geometry for auralization. Detected values are shown in Set Global Room Geo input boxes. **Note:** Without room analyser or Set Global Room geometry active, reflections are generated based on fixed values fitting the room preset.
Room Boundaries	Select which objects are analysed as room boundaries.
Analyser Update Time	Set the Update-time (in seconds) between two raycasts. (0.1 s ⇔ 10.0 s) **Note:** Faster Update-time lead to more performance load.
Debug Room Analyser (Gizmos)	Visualise raycasting for room analyzer. Visible in Scene Window.

	Note: Use to control the objects detected as room boundaries.
Set Global Room Geometry	Enable to set Room Geometry manually. **Note:** Set room geometry manually to the desired values.
UP / DOWN	Listener's distance to the ceiling / ground in meter.
FRONT / BACK	Listener's distance to the front / rear wall in meter. Value modifies delay and level of reflexion from front / rear direction.
LEFT / RIGHT	Listener's distance to the left or right wall in meter. Value modifies delay and level of reflexion from 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 a *dearVR reverb* plugin instances within the Unity Audio Mixer. This way, 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 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 room preset and adjust reverb parameter.

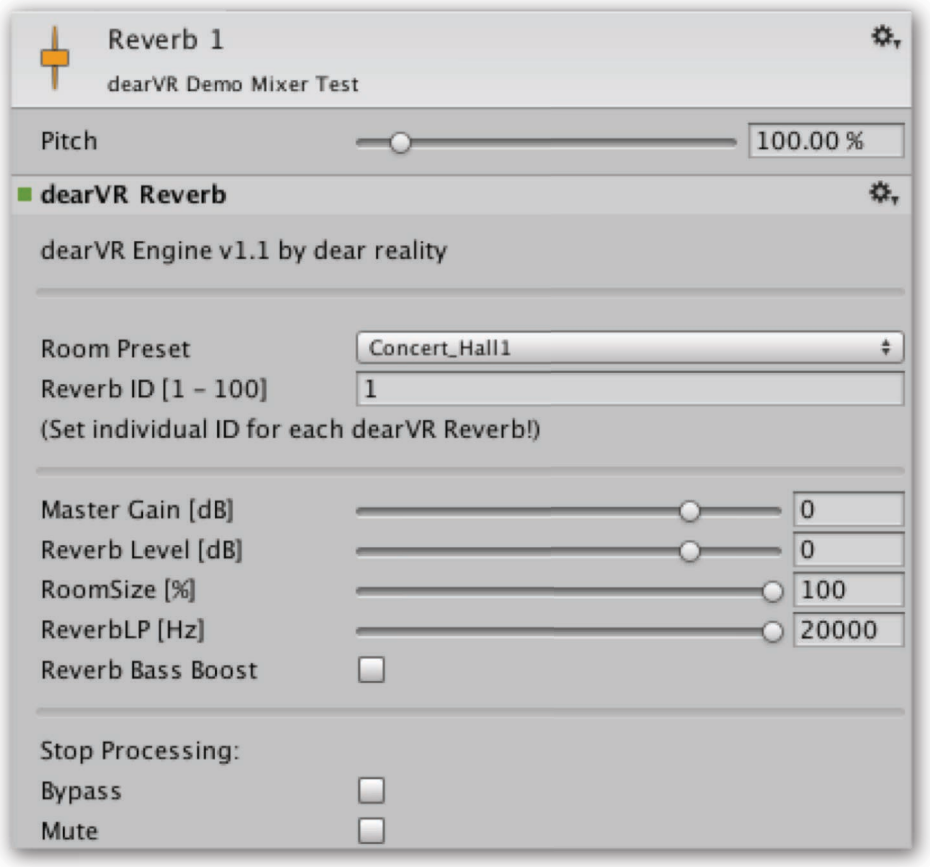

Illustration 4.3 – dearVR Reverb Audio Mixer Plugin

Room Preset	Menu to select a Reverb Room preset. (Refer to 5.6 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. Set 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 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 larger 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.	**Musicproduction**	**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. dearVR API

To control the dearVR properties via 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;

    myDearVRSource.InputChannel = 1.0f;


    // 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;
        }
    }
}
```

7. Mobile Development

iOS & Android:

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

2. 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) ; }

8. Troubleshooting

1. Audible dropouts in consequence of performance issues.
Please check DSP load in Stats Window. Reduce 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 Performace 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 advices given to performance issues.

4. Function *AudioMixerSnapshot.TransitionTo*
Transitions between snapshots are a great feature within Unity. However it is not possible to change 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 *4.1.3 – Settings*) the *Reverb Zone* graph is responsible for the Reverb Level over distance (see *Illustration 4.1.6 – Audio Source Graph).*

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

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

9. Changelog

dearVR v1.2.1

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

dearVR v1.2.0

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

dearVR v1.1.1

WARNING: Due to changes in Unity's (5.3.6+) internal audio engine, left and right channels have to be summed. So the value 0.0 is fixed now and the parameter is depricated. We recommend to recheck your mix and work with mono files from now on.

- Fixed input channel paremter 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 v1.1

- Changed Distance Correction range to [0.01 m - 10 m]
- Changed Occlusion default to 0.6
- Changed Occlusion Update Intervall default to 0.2
- Changed Occlusion Raycast 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

10. Appendix

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 UG
Oberbilker Allee 53
40223 Düsseldorf

Caution

Using headphones requires responsible listening. Damage to hearing occurs when listen 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 to 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 UG will in any event not be liable for any damage to hearing caused by loud sounds.