onAirVR

# onAirVR Server for Unity
# User Manual

Version 1.2.1

# TABLE OF CONTENTS

# INTRODUCTION

onAirVR makes a mobile VR device act as a wireless VR HMD for a desktop by streaming video/audio which are rendered in realtime on the desktop to onAirVR mobile apps. This document describes how to implement onAirVR desktop contents on Unity game engine using onAirVR Server for Unity.

# SYSTEM REQUIREMENTS

## Hardware

- Desktop
  - NVIDIA graphics card **(Kepler architecture or later)**
- Mobile
  - Samsung GearVR

## Software

- Desktop
  - Windows 7 or later
  - **Unity 5.6.x or higher**
  - NVIDIA CUDA Toolkit 8.0
    https://developer.nvidia.com/cuda-downloads
  - The latest NVIDIA Graphics Driver
    **You SHOULD update the graphics driver after installing CUDA toolkit because CUDA toolkit installation may include an older version of the graphics driver.**
- Mobile
  - Android 5.0 (Lollipop) or higher

# QUICK START

1. Put an **AirVRCameraRig** prefab onto a suitable place (e.g. the head position of a player character).

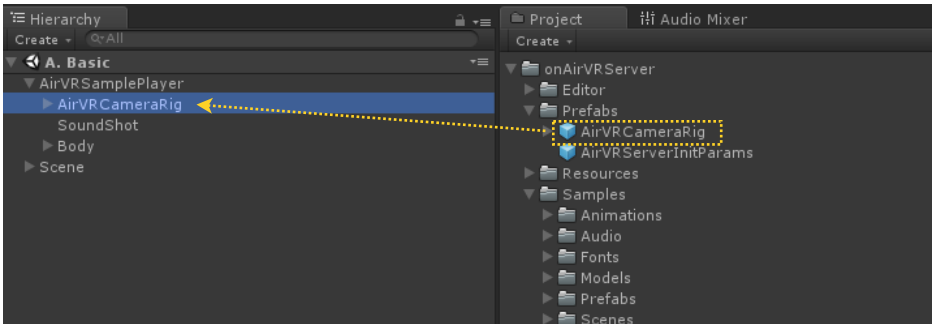



Figure 1. Put an AirVRCameraRig prefab under the player object.

2. Play your project in the editor.
3. Launch onAirVR app on your phone then combine the phone with your GearVR.
4. Enter the IP address of your desktop and tap the Apply button.

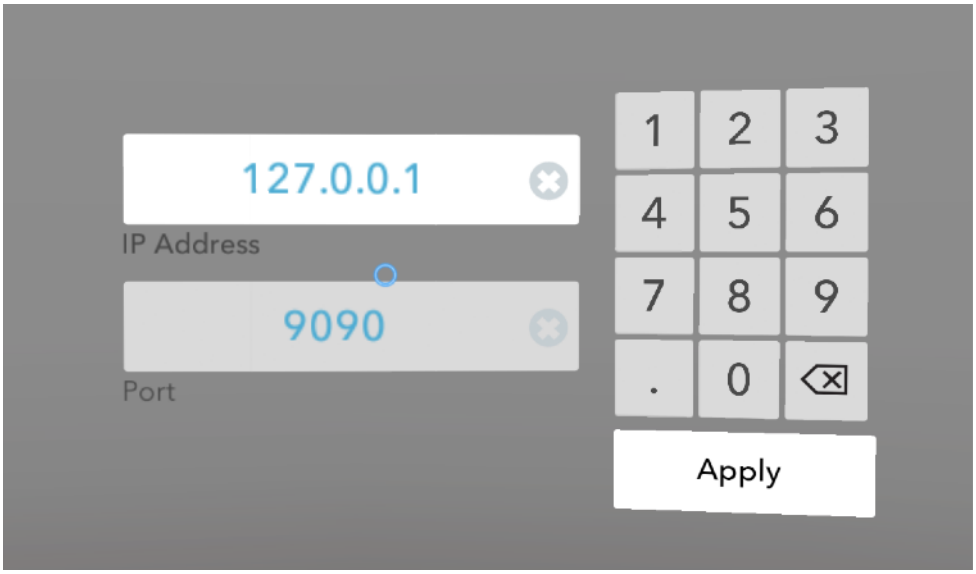



Figure 2. Settings scene of onAirVR app

5. Enjoy your scene!

# PROGRAMMING GUIDE

## How onAirVR Server Works

When an onAirVR server application is launched and the first scene is loaded, the first awaken AirVRCameraRig instantiates an AirVRServer instance and starts it up. And each AirVRCameraRig registers itself to AirVRCameraRigManager in the scene (If no AirVRCameraRigManager exists in the scene, one instance of it is automatically instantiated). Then,

1. When the onAirVR client app on a mobile VR device connects to the onAirVR server application,
2. AirVRServer establishes a session and informs AirVRCameraRigManager.
3. AirVRCameraRigManager then finds an available AirVRCameraRig, and
4. Binds the AirVRCameraRig to the session.
5. Data from the client - such as the HMD orientation, input device values, etc. - are applied to the AirVRCameraRig through the session.
6. Meanwhile AirVRCameraRig renders video frames using child Unity cameras then encodes and sends the video frames back to the client.

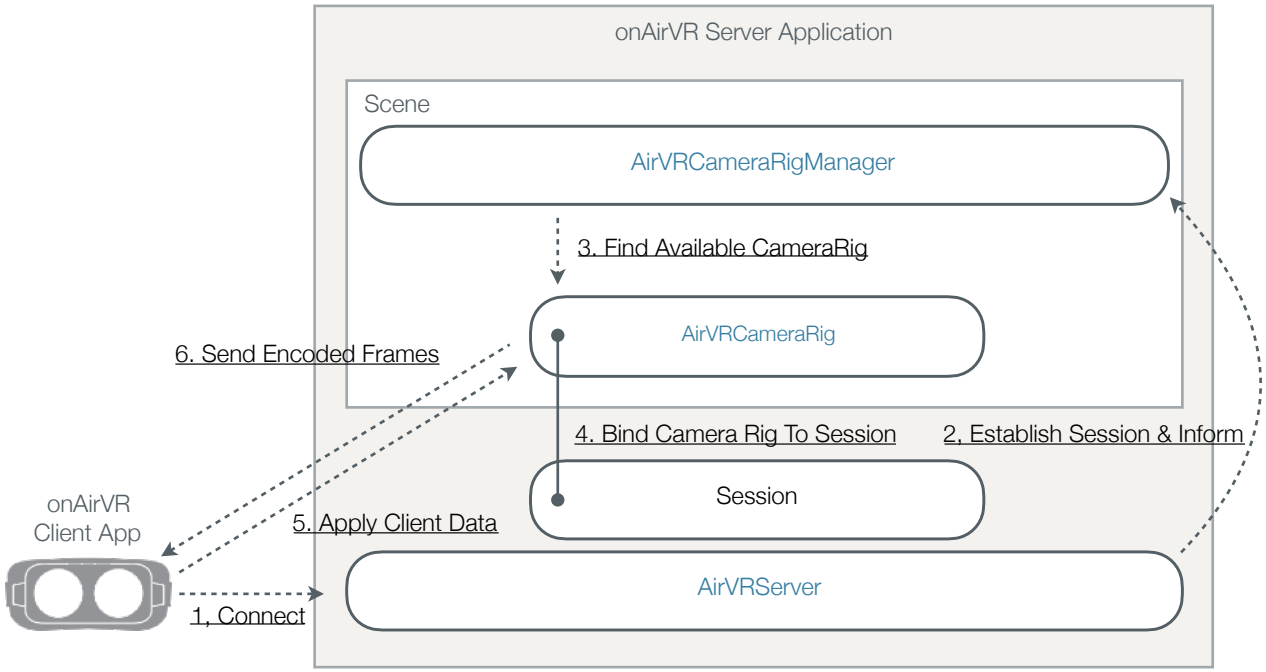



Figure 3. This diagram describes how onAirVR components interact each other.

If you load a new scene containing AirVRCameraRigs,

1. AirVRCameraRigs in the old scene are unbound from the current sessions, then
2. AirVRCameraRigManager tries to bind AirVRCameraRigs in the new scene to the sessions.

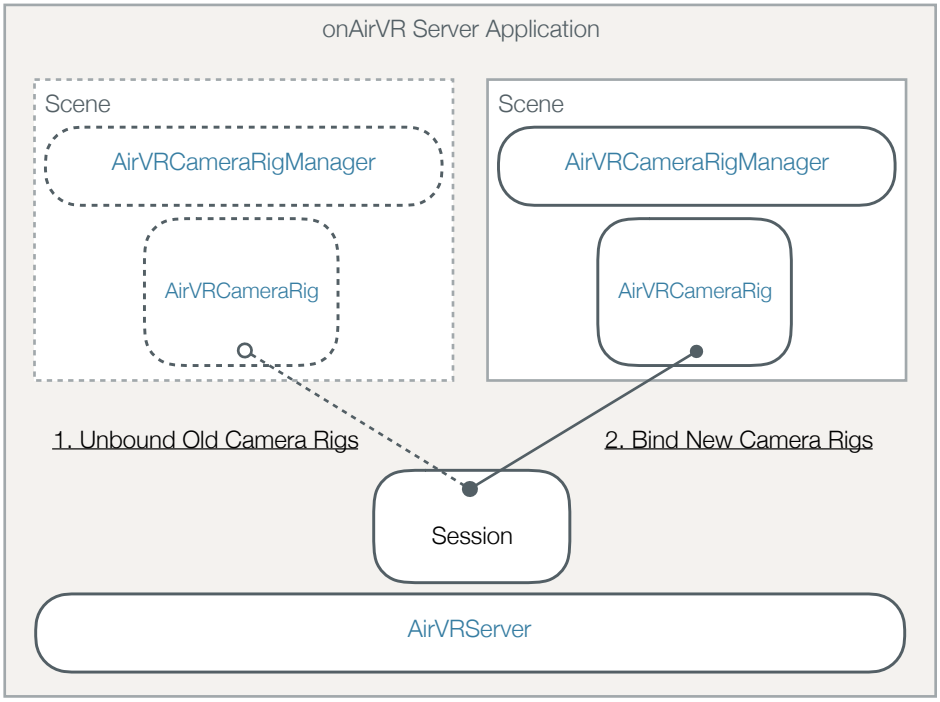



Figure 4. When loading a new scene, AirVRCameraRig bindings are transferred.

## Server Configuration

You can override network configuration and video encoding parameters by putting an AirVRServerInitParams on the scene where AirVRServer will be started up. Then the AirVRServer replace the default configuration to one of the AirVRServerInitParams. Please see “References” section for each field for detail.

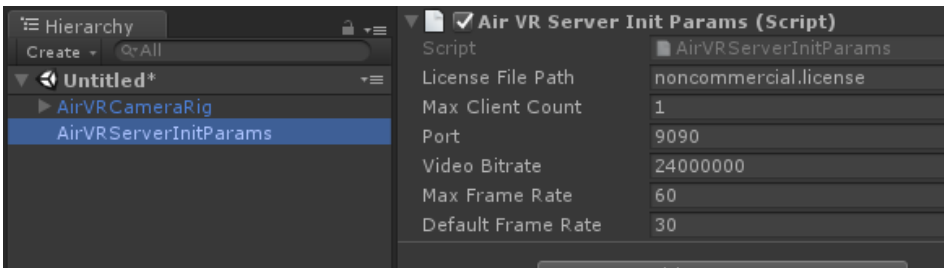



Figure 5. AirVRServerInitParams is put on a scene.

## Multiple Players

onAirVR server acts like a video streaming server which streams realtime-rendered video frames to clients. So it's possible that two or more clients are connecting and playing simultaneously. To make a scene with multiple players, you just need to :

1. Put two or more AirVRCameraRig instances in the scene, and
2. Override the maximum client count to two or more using AirVRServerInitParams.

Then when a session is established for a client, AirVRCameraRigManager finds one of available AirVRCameraRigs in the scene randomly then binds it to the session.
Or if you implement AirVRCameraRigManager.EventHandler, AirVRCameraRigManager requests you to select one of AirVRCameraRigs through AirVRCameraRigManager.EventHandler.AirVRCameraRigWillBeBound().

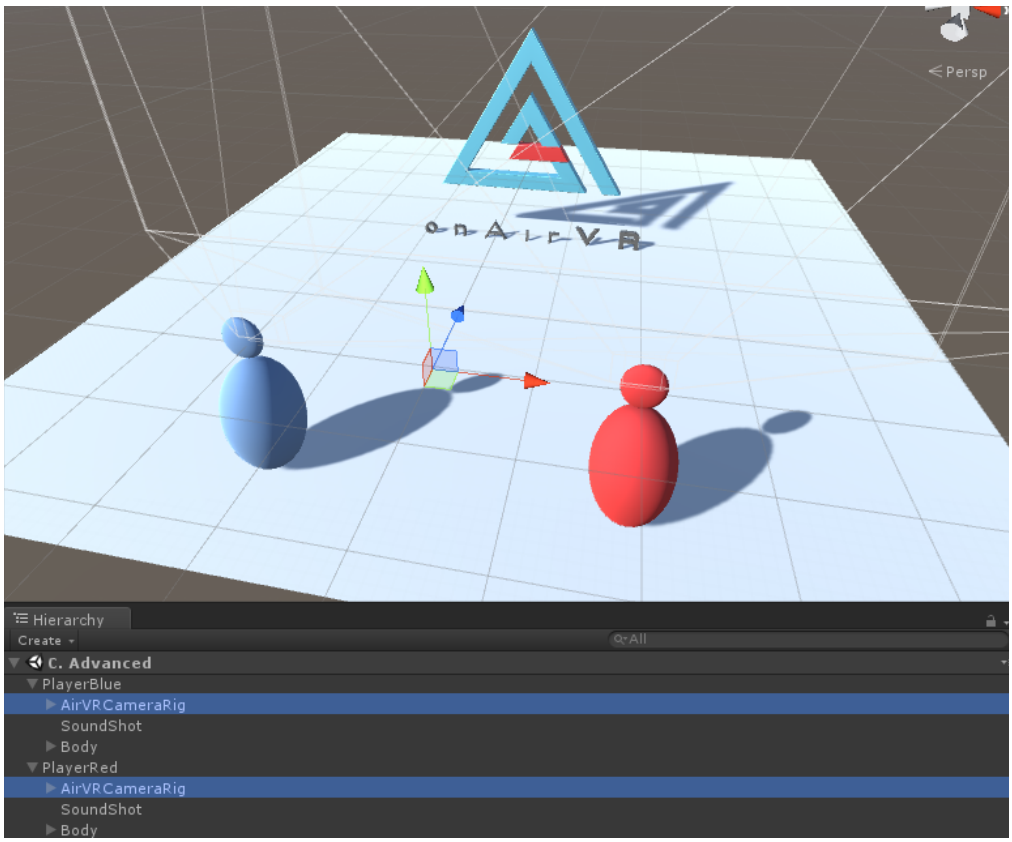



Figure 6. Two AirVRCameraRigs are instantiated in a scene to allow two players.

Note
There is a limitation on the number of encoding sessions depending on your graphics card.
For example, NVIDIA Geforce graphics cards allow up to two encoding sessions due to licensing restrictions.
In this case you must not make your content available for more than two clients simultaneously.

## Event Handling

There are two main components where events you might be interested in are occurred - AirVRServer and AirVRCameraRigManager. If you would like to do something for the events you need to

1. Implement AirVRServer.EventHandler interface and set to AirVRServer.Delegate, and/or
2. Implement AirVRCameraRigManager.EventHandler interface and set to AirVRCameraRigManager.managerOnCurrentScene.Delegate.

Please see “References” section for detail.

```
void Awake() {
    AirVRServer.Delegate = this;
}

public void AirVRServerFailed(string reason) {
    Debug.Log(reason);
}

public void AirVRServerClientConnected(IntPtr clientHandle) { }

public void AirVRServerClientDisconnected(IntPtr clientHandle) { }
```

Figure 7. An example of AirVRServer.EventHandler implementation

```
void Awake() {
    AirVRCameraRigManager.managerOnCurrentScene.Delegate = this;
}

public void AirVRCameraRigWillBeBound(AirVRClientConfig config, List<AirVRCameraRig> availables, out AirVRCameraRig selected) {
    selected = availables.Count > 0 ? availables[0] : null;
}

public void AirVRCameraRigActivated(AirVRCameraRig cameraRig) {}

public void AirVRCameraRigDeactivated(AirVRCameraRig cameraRig) {}

public void AirVRCameraRigHasBeenUnbound(AirVRCameraRig cameraRig) {}
```

Figure 8. An example of AirVRCameraRigManager.EventHandler implementation

## Input

Using AirVRInput class, you can get the values of input devices of a connected mobile HMD bound to an AirVRCameraRig. As you can see in “References” section, you can use AirVRInput class in the same manner with UnityEngine.Input except that AirVRInput methods require an AirVRCameraRig as an argument. Currently onAirVR mobile app supports GearVR Touchpad, Xbox Controller and GearVR Controller and the below figures describe input bindings of each input device. Also note that you can get the pose of input devices which pose is tracked by sensor - HMD and GearVR Controller - using AirVRInput.GetPointerPositionAndOrientation().

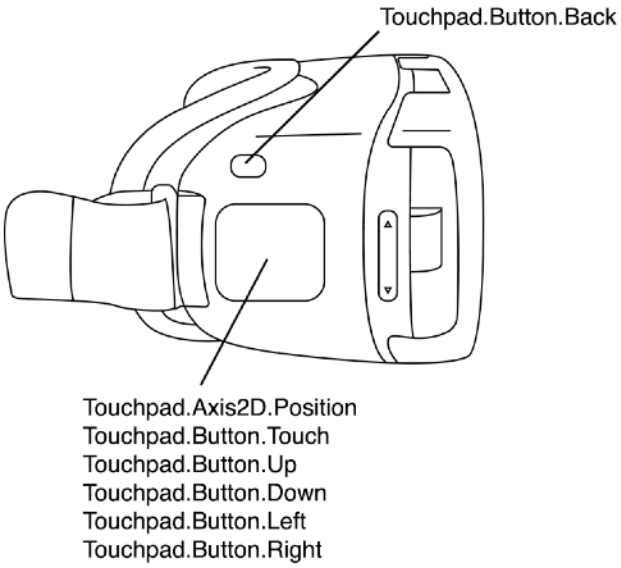



Figure 9. Axes and buttons of GearVR Touchpad

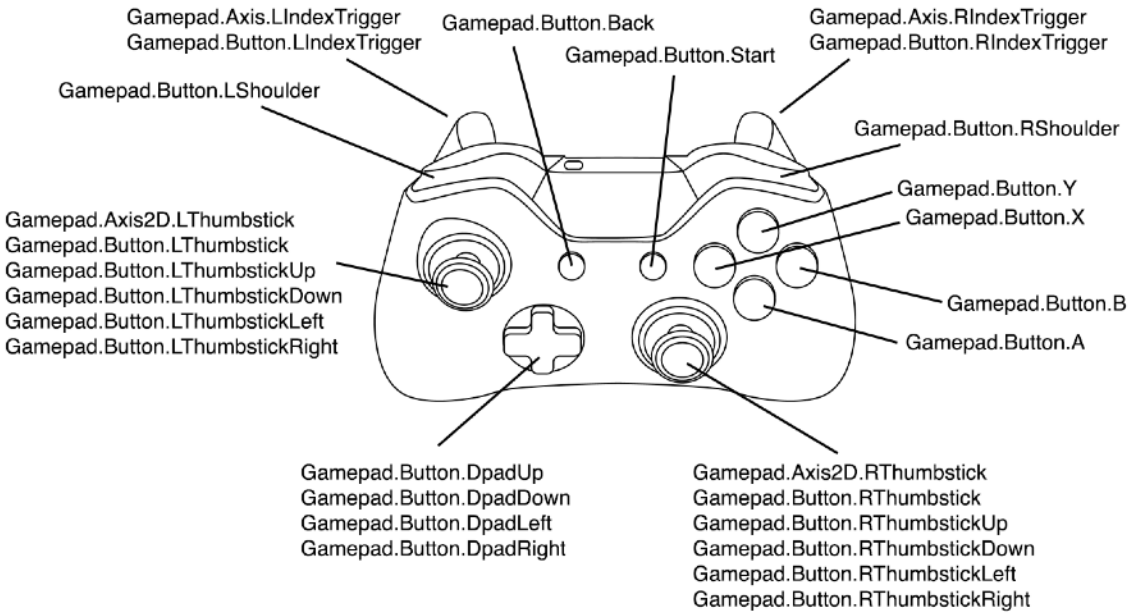



Figure 10. Axes and buttons of Xbox Controller

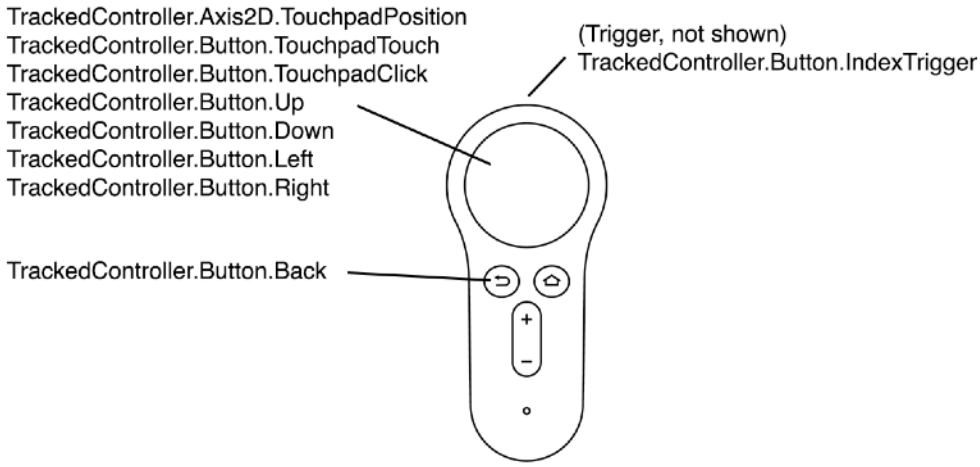



Figure 11. Axes and buttons of GearVR controller

## Audio

AirVRServerAudioOutputRouter routes “stereo” audio rendered by Unity audio engine to connected clients. It must be attached to an GameObject on which a Unity’s AudioListener is attached.
(Please see AirVRServerAudioOutputRouter component in **CenterEyeAnchor** of **AirVRCameraRig** prefab.)

There are several options for input and output.

- Input
  - AudioListener : routes stereo audio heard by Unity’s AudioListener to clients
  - AudioPlugin : routes stereo audio passing through “AirVR Server Audio Output” audio plugin in a Unity’s AudioMixer to clients
- Output
  - All : broadcast audio to all of connected clients
  - One : routes audio to a specific client

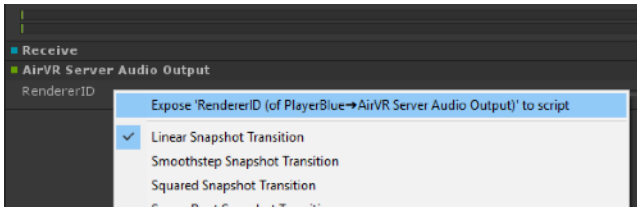
In **AirVRCameraRig** prefab, AirVRServerAudioOutputRouter routes AudioListener input to All output by default. Please see “C. Advanced” sample scene for an example which uses AudioPlugin as input.

For advanced audio application, you might want to see the code of AirVRServerAudioOutputRouter, in which it takes raw audio data from Unity audio engine then send them to clients using AirVRServer.SendAudioFrame(). You can use this method if you want to send your own raw audio directly, for example in the case you are using a 3rd-party or your own audio engine.

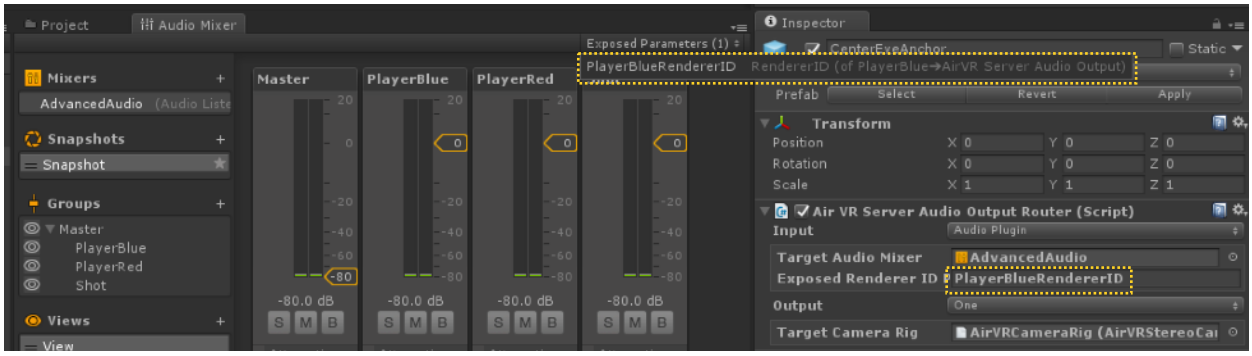
Note

“AirVR Server Audio Output” audio plugin uses “audio renderer ID” to send audio data to AirVRCameraRig. When you set AudioPlugin as input, you must (1) expose RendererID parameter of the plugin then (2) set it to exposedRendererIDParameterName field of AirVRServerAudioOutputRouter.

(1)




(2)



# BUILD

onAirVR Server for Unity supports only 64bit architecture. So you must set 'Architecture' in Build Settings to 'x86_64' before you build.

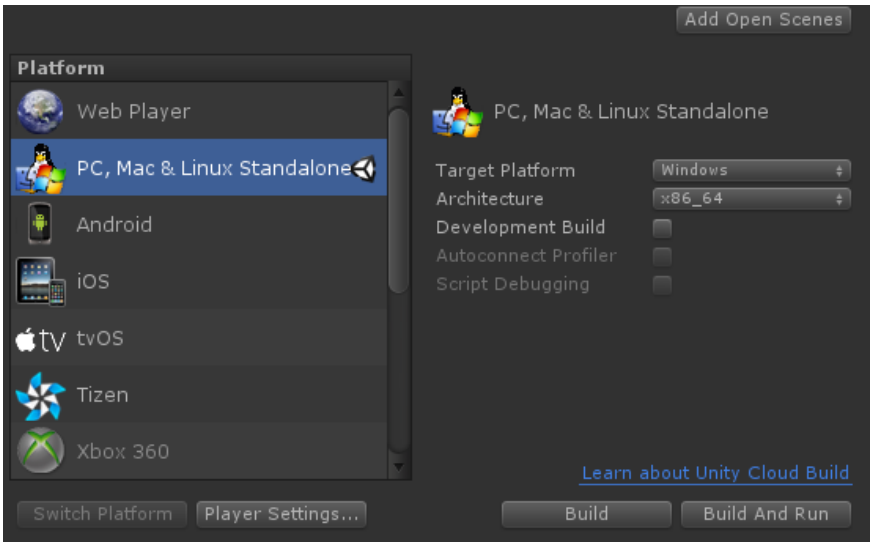



Figure 12. 'Architecture' must be set to x86_64 in Build Setting.

And you have to distribute the built application with dependent libraries as below :

- **cudart64_80.dll**
  - *C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\bin\cudart64_80.dll*

Finally, an onAirVR server license file must be in the directory the executable is in. You can use *noncommercial.license* - which is in *Assets/onAirVRServer/Editor/Misc/* - included in our package, only for noncommercial purpose.

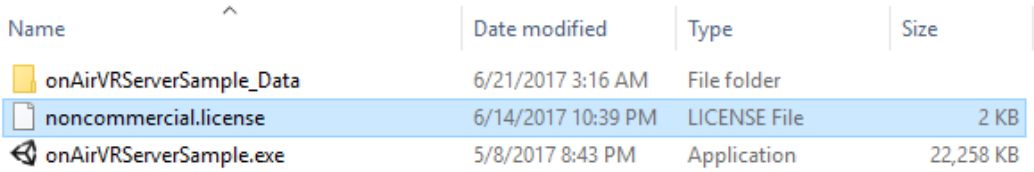



Figure 13. You can copy *noncommercial.license* file into the directory the executable is in.

# BEST PRACTICES

## Setting User ID In onAirVR Client App

There might be cases where an onAirVR server application need to know actually what onAirVR client is connected. For example if you plan to integrate an outside-in positional tracking system into your onAirVR application for multiple players, you need to match an onAirVR client to the rigid body which tracks the HMD of that client in the positional tracking system. We provide the way to assign a User ID to an onAirVR client to solve such problems.

This is how to assign a User ID in onAirVR app :

1. In Settings, swipe up the Touchpad while focusing IP Address input field, then User ID input field appears.
2. Enter a number as a User ID which you want to assign, then tap Apply button.
3. You can get the User ID from AirVRClientConfig.userID property.

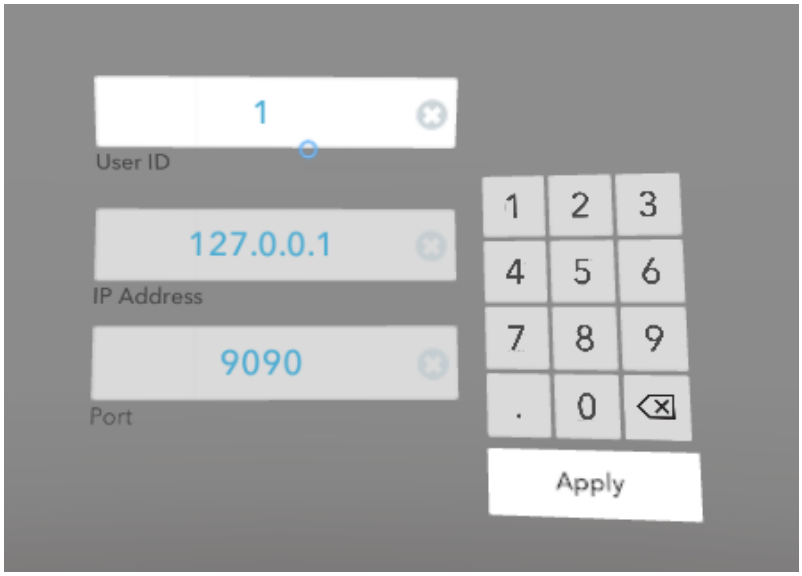



Figure 14. User ID input field in Settings of onAirVR app

## Correcting Yaw Difference When Using An External Positional Tracking System

If you are using an external outside-in positional tracking system to track the position and orientation of an onAirVR client, there might be an issue about yaw difference between the onAirVR client's orientation driven from its own inertial sensors and the orientation tracked by the positional tracking system. The yaw difference always occurs because most mobile VR HMDs reset its orientation to identity on start, no matter how they pose at that time in real world. We suggest the simplest and effective way to solve this issue :

In AirVRCameraRigManager.EventHandler.AirVRCameraRigActivated(AirVRCameraRig cameraRig),

1. Recenter the pose of cameraRig by calling cameraRig.RecenterPose().
2. Calculate the yaw difference between the orientation from your positional tracking system and the orientation of the transform of cameraRig.
3. Apply the yaw difference to the transform of cameraRig (or one of its parents).

There might be gradual yaw drift in the onAirVR client as times go on. Unfortunately there isn't any effective correction method working gradually in the case like onAirVR yet. But thanks to the excellent performance of GearVR, there is rarely yaw drift even if playing a significant time without correction.

# REFERENCES

## AirVRServer

Fundamental component responsible for onAirVR server startup/shutdown, video encoder and network configuration and client connection management. It also includes methods to send audio.

| Static Variables | | |
|---|---|---|
| Delegate | AirVRServer.EventHandler | instance of interface through which events are notified |

**Static Functions**

bool GetAudioRendererID(AirVRCameraRig cameraRig, ref int rendererID)

gets the renderer ID of *cameraRig* for audio plugin. (See “Programming Guide - Audio” section for detail.)

- Return Value
  - true if succeeded to get the renderer ID
- Parameters
  - cameraRig : camera rig of which you want to get the renderer ID
  - rendererID : out parameter for renderer ID

void SendAudioFrame(AirVRCameraRig cameraRig, float[] data, int sampleCount, int channels, double timestamp)
void SendAudioFrameToAllCameraRigs(float[] data, int sampleCount, int channels, double timestamp)

sends raw audio data directly to the client of a camera rig.

- Parameters
  - cameraRig : camera rig to which you want to send audio data
  - data : raw audio data in which each sample is represented in float
  - sampleCount : the number of samples in data
  - channels : the number of channels. the first two channels are considered as left and right channels.
  - timestamp : the timestamp of audio data (in seconds).

## AirVRServerInitParams

Overrides onAirVR server configuration if this component exists in the scene where AirVRServer is started up. Please read “Programming Guide - Server Configuration” section to see how to use in detail.

| Variables | | |
|---|---|---|
| licenseFilePath | string | the path of an onAirVR server license file which the built executable will use. (In editor, *Assets/onAirVRServer/Editor/Misc/noncommercial.license* is used always. See “Build” section for detail.) |
| maxClientCount | int | the maximum number of clients which can be connected simultaneously |
| port | int | onAirVR server port number |
| videoBitrate | int | bitrate in which to encode video for each client (bps) |
| maxFrameRate | float | maximum frame rate in which to encode video for each client (fps) |
| defaultFrameRate | float | default frame rate in which to encode video for each client (fps) |

## AirVRServer.EventHandler

Interface through which AirVRServer events are notified. You might implement this interface and set to AirVRServer.Delegate to do something for events on server errors or client connections. The below figure describes what events are occurred when in AirVRServer.

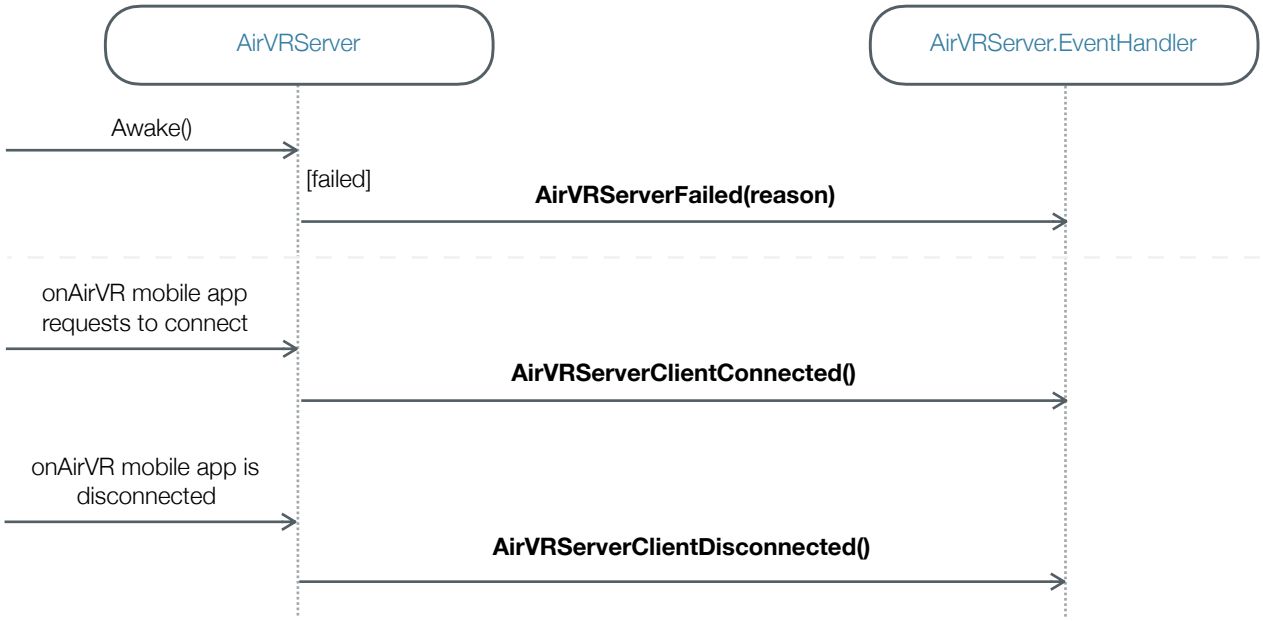



Figure 15. This sequence diagram describes what AirVRServer events are occurred when, through AirVRServer.EventHandler.

| Public Functions | |
|---|---|
| void AirVRServerFailed(string reason) | called when onAirVR server fails to startup<br>• Parameters<br>- reason : reason why startup is failed |
| void AirVRServerClientConnected (IntPtr clientHandle) | called when a client is connected.<br>• Parameters<br>- clientHandle : the connection handle of the connected client |
| void AirVRServerClientDisconnected (IntPtr clientHandle) | called when a connected client is disconnected.<br>• Parameters<br>- clientHandle : the connection handle of the client |

## AirVRCameraRigManager

Is responsible for binding AirVRCameraRigs in a scene to the sessions of connections to onAirVR clients. It is guaranteed that there is an AirVRCameraRigManager instance in a scene as long as the scene contains more than one AirVRCameraRig.

| Static Variables | | |
|---|---|---|
| Delegate | AirVRCameraRigManager.EventHandler | instance of interface through which events are notified |
| managerOnCurrentScene | AirVRCameraRigManager | the instance on the current scene |

## AirVRCameraRigManager.EventHandler

Interface through which AirVRCameraRigManager events are notified. You might implement this interface and set to AirVRCameraRigManager.Delegate to do something for events occurred for AirVRCameraRig. The below figure describes what events are occurred in AirVRCameraRigManager.

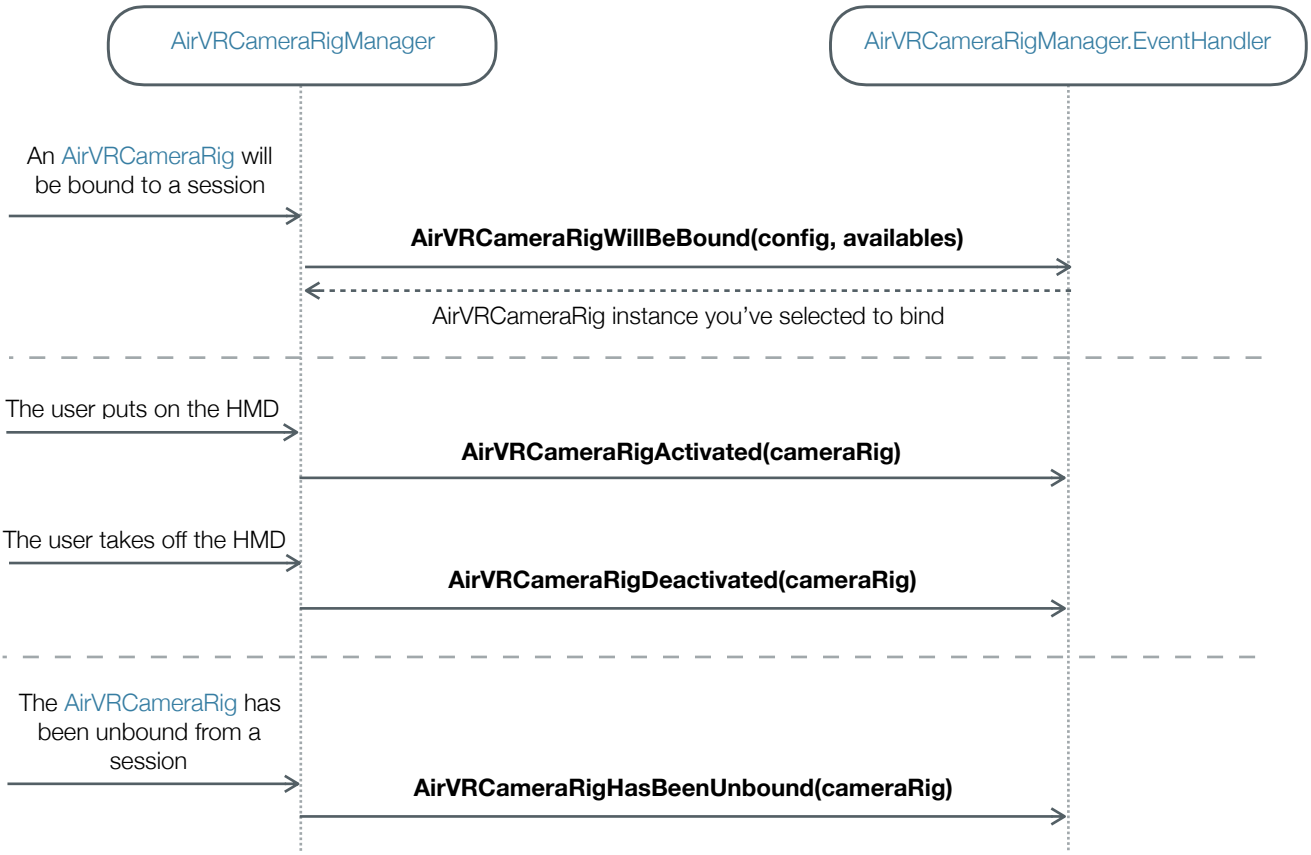



Figure 16. This sequence diagram describes what AirVRCameraRigManager events are occurred when, through AirVRCameraRigManager.EventHandler.

| Public Functions | |
|---|---|
| void AirVRCameraRigWillBeBound (AirVRClientConfig config, List<AirVRCameraRig> availables, out AirVRCameraRig selected) | called just before an AirVRCameraRig is bound to a session of AirVRServer. You must choose an instance of AirVRCameraRig and assign it to parameter *selected*, which might be one in parameter *availables* or one you've just instantiated in this method. Also, you can change the configuration for the client through parameter *config*.<br>• Parameters<br>- config : configuration for the connected client<br>- availables : AirVRCameraRig instances in your scene which are not bound to any session yet<br>- selected : the AirVRCameraRig instance you've selected to bind. Assign *null* if you want to reject this binding. |
| void AirVRCameraRigActivated (AirVRCameraRig cameraRig) | called when a user puts on the HMD.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to the client of the user |
| void AirVRCameraRigDeactivated (AirVRCameraRig cameraRig) | called when a user takes off the HMD.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to the client of the user |
| void AirVRCameraRigHasBeenUnbound (AirVRCameraRig cameraRig) | called just after an AirVRCameraRig has been unbound from a session.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance which was bound to the session |

## AirVRClientConfig

Configuration values requested from a connected client, such as the arrangement of eyes, field of view (FOV), frame rate of video, etc. You can change some configuration values on AirVRCameraRigManager.EventHandler .AirVRCameraRigWillBeBound() callback.

| Variables | | |
|---|---|---|
| type | AirVRClientType | the type of the client (read only). Always "Stereoscopic" currently. |
| videoWidth | int | the width of video to send to the client (read only) |
| videoHeight | int | the height of video to send to the client (read only) |
| framerate | float | frame rate of encoded video to send to the client (read only) |
| fov | float | vertical field of view of cameras (read only) |
| eyeCenterPosition | Vector3 | offset from the root to CenterEyeAnchor in AirVRCameraRig |
| ipd | float | distance between the left and right eyes |
| userID | int | user ID which is specified in onAirVR client app (read only). Read "Best Practice" section for detail. |
| userData | object | store what you want to attach to the session the AirVRCameraRig is bound to. |

## AirVRCameraRig

Abstract base class for manipulating cameras which render video frames to send to onAirVR clients.

| Variables | | |
|---|---|---|
| type | AirVRClientType | the type of AirVRCameraRig (read only). Always “Stereoscopic” currently. |
| isBountToClient | bool | true if this is bound to a client through a session currently (read only). |

| Public Functions | |
|---|---|
| AirVRClientConfig GetConfig() | returns the configuration values requested from the client this camera rig is bound to. |
| void AdjustBitrate(uint bitrateInKbps) | adjusts encoding bit rate of video frames to send to the client, in “kbps”. (1 kbps = 1000 bps) |
| void RecenterPose() | recenters the pose of the client. |
| void Disconnect() | disconnects from the connected client this camera rig is bound to. |

## AirVRStereoCameraRig (inherits from AirVRCameraRig)

A set of cameras which render stereoscopic video frames to send to a client of “Stereoscopic” type.

| Variables | | |
|---|---|---|
| leftEyeCamera | Camera | Camera which acts as the left eye. This renders the left side of a video frame for stereoscopic video type. |
| rightEyeCamera | Camera | Camera which acts as the right eye. This renders the right side of a video frame for stereoscopic video type. |
| leftEyeAnchor | Transform | Transform of leftEyeCamera |
| centerEyeAnchor | Transform | Transform of centerEyeCamera |
| rightEyeAnchor | Transform | Transform of rightEyeCamera |

## AirVRInput

Provides methods to get values of input devices - including GearVR Touchpad, Xbox Controller and GearVR Controller - of the client an AirVRCameraRig are bound to. Please see “Programming Guide - Input” section to check what each button or axis is bound to a control in an input device.

| Static Functions | |
|---|---|
| bool IsDeviceAvailable(AirVRCameraRig cameraRig, AirVRInput.Device device) | returns true if an input device is available on the client which cameraRig is bound to.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to a client<br>- device : the type of the input device<br>✓ AirVRInput.Device<br>- HeadTracker (just tracks the head pose)<br>- Touchpad<br>- Gamepad<br>- TrackedController |
| void GetPointerPositionAndOrientation (AirVRCameraRig cameraRig, AirVRInput.Device device, out Vector3 worldPosition, out Quaternion worldOrientation) | gets the “world” position and orientation of an input device which acts as a pointing device.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to a client<br>- device : the type of the input device. HeadTracker and TrackerController are available only.<br>- worldPosition / worldOrientation : out parameters for world position and orientation of the input device |
| Vector2 Get(AirVRCameraRig cameraRig, AirVRInput.Touchpad.Axis2D axis)<br>bool Get(AirVRCameraRig cameraRig, AirVRInput.Touchpad.Button button)<br>Vector2 Get(AirVRCameraRig cameraRig, AirVRInput.Gamepad.Axis2D axis)<br>float Get(AirVRCameraRig cameraRig, AirVRInput.Gamepad.Axis axis)<br>bool Get(AirVRCameraRig cameraRig, AirVRInput.Gamepad.Button button)<br>Vector2 Get(AirVRCameraRig cameraRig, AirVRInput.TrackedController.Axis2D axis)<br>bool Get(AirVRCameraRig cameraRig, AirVRInput.TrackedController.Button button) | returns the value of an axis or a button of an input device of the client which *cameraRig* is bound to.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to a client<br>- axis/button : one of axes/buttons of an input device |

| Static Functions | |
|---|---|
| bool GetDown(AirVRCameraRig cameraRig, AirVRInput.Touchpad.Button button)<br>bool GetDown(AirVRCameraRig cameraRig, AirVRInput.Gamepad.Button button)<br>bool GetDown(AirVRCameraRig cameraRig, AirVRInput.TrackedController.Button button) | returns true if a button of an input device of the client which *cameraRig* is bound to is pressed during the frame.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to a client<br>- button : one of buttons of an input device |
| bool GetUp(AirVRCameraRig cameraRig, AirVRInput.Touchpad.Button button)<br>bool GetUp(AirVRCameraRig cameraRig, AirVRInput.Gamepad.Button button)<br>bool GetUp(AirVRCameraRig cameraRig, AirVRInput.TrackedController.Button button) | returns true if a button of an input device of the client which *cameraRig* is bound to is released during frame.<br>• Parameters<br>- cameraRig : AirVRCameraRig instance bound to a client<br>- button : one of buttons of an input device |

## AirVRServerAudioOutputRouter

Routes “stereo” audio rendered by Unity audio engine to connected clients. Must be attached to a GameObject on which Unity’s AudioListener is attached. Please read “Programming Guide - Audio” section how it works.

| Variables | | |
|---|---|---|
| input | Input | audio source to send to connected clients<br>• Input<br>- AudioListener<br>· takes stereo audio heard by Unity’s AudioListener as input<br>- AudioPlugin<br>· takes stereo audio passing through “AirVR Server Audio Output” audio plugin as input |
| output | Output | destination to which onAirVR server will send audio<br>• Output<br>- All<br>· broadcast audio to all connected clients<br>- One<br>· send audio only to a specific client |
| targetAudioMixer | UnityEngine.Audio.AudioMixer | Unity’s AudioMixer containing “AirVR Server Audio Output” audio plugin. Used only when input is “AudioPlugin”. |
| exposedRendererIDParameterName | string | exposed parameter name of RendererID parameter of “AirVR Server Audio Output” audio plugin in targetAudioMixer. Used only when input is “AudioPlugin”. |
| targetCameraRig | AirVRCameraRig | AirVRCameraRig instance bound to a client to which want to send audio. Used only when output is “One”. |

# TROUBLESHOOTING

### Q. DllNotFoundException occurs in Unity console.

It is due to missing libraries on which onAirVR server depends. Check out the system requirements : The latest NVIDIA graphics driver, CUDA Toolkit 8.0

### Q. onAirVR server fails with reason “License file not found”

Check if *noncommercial.license* file is in your project directory. (the parent directory of *Assets/*)

### Q. onAirVR mobile app keeps failing to connect.

Windows Firewall might block inbound connections to Unity editor or the onAirVR server application you built. Please check Windows Firewall settings and set to allow connections.

### Q. Succeeded to connect, but the display of onAirVR mobile app is black or green.

Update NVIDIA graphics driver of your desktop to the latest one.
(CUDA Toolkit installation might force to install an older version of NVIDIA graphics driver. If that happens you must update the driver after the installation.)

### Q. I can see my scene on my onAirVR mobile app, but video frames are often distorted significantly or dropped.

If your computer runs your contents well enough, mostly it is due to the wireless network condition. There are several things you need to check.

1. We recommend to use a 5GHz Wi-Fi router. It provides more stable data communication than 2.4GHz according to our experience in many exhibitions.
2. Change the channel of the Wi-Fi on the router setting. Do not use channels your neighbors’ routers are already using.
3. Hide the Wi-Fi SSID on the router setting. In crowded environments, it helps the router to avoid interferences from people’s mobiles which are seeking an available Wi-Fi around them.