# Migrating Unity Applications from Oculus to OSVR

Greg Aring[1] and Yuval Boger[2], Mar 12th, 2015

## What is OSVR?

OSVR™ is an open-source software platform for VR/AR applications.

OSVR provides an easy and standardized way to discover, configure and operate hundreds of devices: VR goggles, position trackers, depth cameras, game controllers and more. OSVR supports multiple operating systems, plugs into leading game engines and is freely available under a permissive Apache 2.0 license.

OSVR was started by experts in gaming and virtual reality and is supported by an ever-growing list of hardware vendors, game studios, universities and software companies.

## Why Migrate?

Developers that created Unity applications designed to run on Oculus products, can gain several advantages by converting them to run on top of OSVR. These migrated applications:

- Support a wide range of VR headsets, including Oculus, OSVR HDK and many others. OSVR automatically configures the field of view and uses device-specific distortion correction shaders where required.
- Gain standardized access to a wide range of peripherals such as eye trackers, gesture cameras.
- Run on an open-source platform that is transparent and easy to enhance.
- Can run on multiple operating systems

## Before Starting

This guide assumes you are able to successfully run the VRDemo_Tuscany.unity scene that comes with the OculusUnityIntegrationTuscanyDemo package. If you are unable to run this, consult the OculusUnityIntegrationGuide.pdf that comes with the Oculus SDK for help.

You will also need to download the following from http://access.osvr.com :

- OSVR-Core Windows binary snapshot (latest 32bit recommended)

---

[1] Greg Aring is a software engineer at Sensics.

[2] Yuval Boger is the CEO of Sensics. Sensics is the founding contributor of OSVR. He can be reached at vrguy@sensics.com

- OSVR-Unity Windows binary assets
- OSVR Oculus Plugin

## Configuring the OSVR Server

The OSVR server will need to be running any time you are using it in your project. You might want to create a shortcut on your taskbar or desktop so you can start the server quickly. We will show you how to configure the server to run with the Oculus Rift so that you can perform step-by-step migration.

The OSVR server (**osvr_server.exe**) is located in the **bin** directory of the OSVR-Core snapshot.

- First, copy **com_osvr_OculusRift.dll**, located in the OSVR Oculus plugin download, to the **osvr-plugins-0** directory of the OSVR-Core snapshot.

When it starts, OSVR server reads from a configuration file which is, by default, **osvr_server_config.json**. You can specify a different configuration file by passing the filename as an argument when running osvr_server from a command line. The configuration file needs to be in the same directory as the OSVR server. The configuration file defines the hardware to be used, the internal routing of information inside the OSVR stack and so forth[3]. There are a few ways to configure the server to run with the Oculus Rift.

1) Copy the contents of **osvr_server_config.oculusrift.sample.json**, located in the OSVR Oculus plugin download, into **osvr_server_config.json** using any text editor. This will overwrite the default configuration, but there is a backup in the bin folder, along with many other example configurations.

2) Copy **osvr_server_config.oculusrift.sample.json** to the bin folder of the OSVR-Core snapshot. You would then run the OSVR server with the command:
"osvr_server osvr_server_config.oculusrift.sample.json**"**

After configuring the OSVR server, the console output when running it should look like this:

[3] For additional information, we suggest reviewing "An Introduction to OSVR" technical white paper

```
[OSUR Server] Using default config file - pass a filename on the command line to
 use a different one.
[OSUR Server] Using config file 'osvr_server_config.json'
[OSUR Server] Constructing server as configured...
[OSUR] Added device: OSUR
[OSUR Server] Loading plugins...
[OSUR Server] Successful plugins:
[OSUR Server]  - com_osvr_OculusRift
[OSUR Server]
[OSUR Server]
[OSUR Server] Instantiating configured drivers...
[OSUR] Added device: com_osvr_OculusRift/OculusRift0
[OSUR Server] Successes:
[OSUR Server]  - com_osvr_OculusRift/OculusRift
[OSUR Server]
[OSUR Server]
[OSUR Server] Triggering a hardware detection...
[OSUR] Performing hardware auto-detection.
[OSUR Server] Registering shutdown handler...
[OSUR Server] Starting server mainloop...
```

## Porting the Unity Project to OSVR

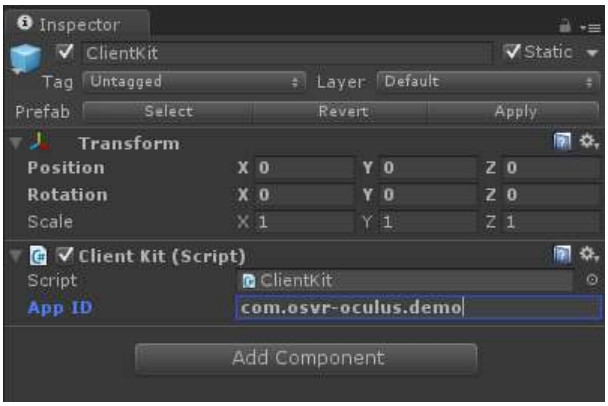
We will start with the Tuscany example project provided in the Oculus SDK.

1) Open VRDemo_Tuscany.unity.

2) Import OSVR-Unity.unitypackage into the project.

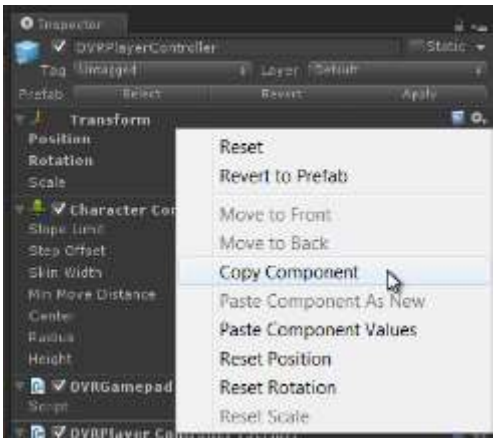
3) Add two prefabs to the scene working:

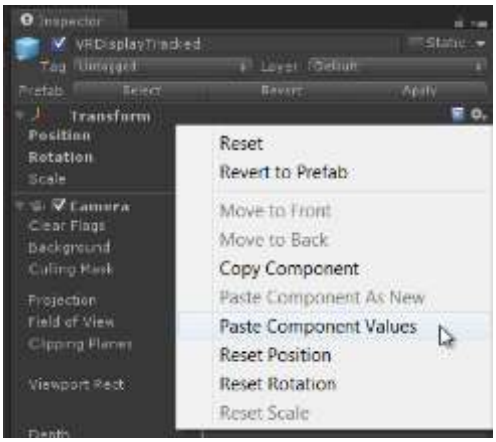
- Drag the **ClientKit** prefab into the scene from the Assets/OSVRUnity/Prefabs folder. OSVR requires one ClientKit object to be in your scene. Select the ClientKit gameobject and examine the ClientKit script in the inspector. In the App ID field, provide a unique name for your application in reverse-domain format, such as "com.osvr-oculus.demo".




- Drag the **VRDisplayTracked** prefab from the Assets/OSVR-Unity/Prefabs folder into the scene. This will be your camera. You might want to align it with the camera already in the scene. To do this, copy the transform component of **OVRPlayerController** and paste the component values onto the transform component of **VRDisplayTracked**. You can now delete or deactivate **OVRPlayerController** from the scene. Note that VRDisplayTracked does not include a player

controller, only a camera.




4) Make sure the OSVR server is running and press Play in the Unity editor.

That's all folks!

## For additional information

There are many ways to keep abreast of OSVR developments:

- Sign up to the OSVR update email on http://access.osvr.com/
- Subscribe to the OSVR/Unity plugin project on Github as well as other OSVR projects
- Visit www.osvr.com

Technical support questions? Email support@osvr.com