# CYBERITH SDK

## Unity Project - Integration Guideline

Cyberith GmbH
Arbeitergasse 25
3130 Herzogenburg
Austria
FN 410899p

**For any question, please contact**

Pierre Mercy
Software Developer
p.mercy@cyberith.com
+43 660 71 76 828

# Table of Contents

# Prerequisites

### Unity3D Game Engine (recommended 5.0.1)

http://unity3d.com/

### Oculus SDK & Runtime (recommended 0.5.0.1)

https://developer.oculus.com/downloads/

### DirectX SDK (June 2010)

https://www.microsoft.com/en-us/download/details.aspx?id=6812

# First Steps

First of all, please note that:

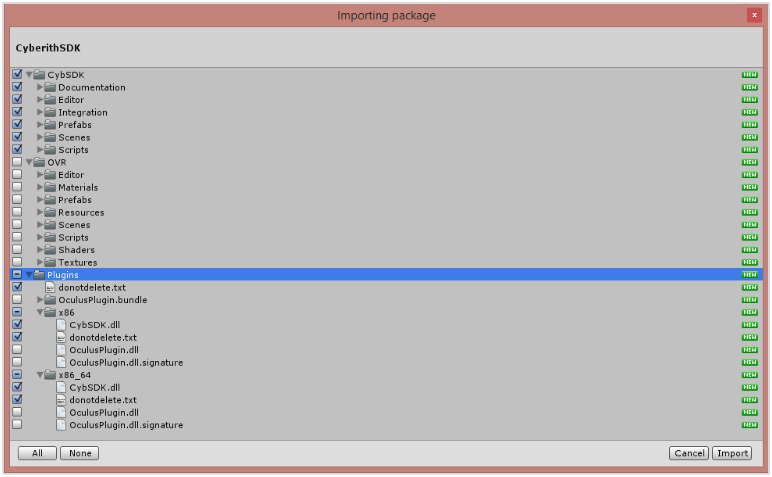
- If you don't have a Virtualizer plugged in, the SDK will detect it and switch automatically to a mouse+keyboard or gamepad controller (Xbox360 gamepad is recommended).
- Later is explained how to switch from coupled or decoupled.

## Setup

The Unity package you received works totally independently so you can import it in an empty Unity project and you should be able to try it using the example scene located in "CybSDK/Scenes/ BaseScene".

If you have already a project with the Oculus Rift SDK:

1. For easier integration, please port your project to at least version 0.4.3 of Oculus SDK.
2. Then, just import from the package the whole CybSDK folder and the Cyberith's parts of the Plugin folder.



Importing package

CyberithSDK

```
☑ ▼ CybSDK
☑   ▶ Documentation
☑   ▶ Editor
☑   ▶ Integration
☑   ▶ Prefabs
☑   ▶ Scenes
☑   ▶ Scripts
□ ▼ OVR
□   ▶ Editor
□   ▶ Materials
□   ▶ Prefabs
□   ▶ Resources
□   ▶ Scenes
□   ▶ Scripts
□   ▶ Shaders
□   ▶ Textures
⊟ ▼ Plugins
☑     donotdelete.txt
□   ▶ OculusPlugin.bundle
⊟   ▼ x86
☑       CybSDK.dll
☑       donotdelete.txt
□       OculusPlugin.dll
□       OculusPlugin.dll.signature
⊟   ▼ x86_64
☑       CybSDK.dll
☑       donotdelete.txt
□       OculusPlugin.dll
□       OculusPlugin.dll.signature
```

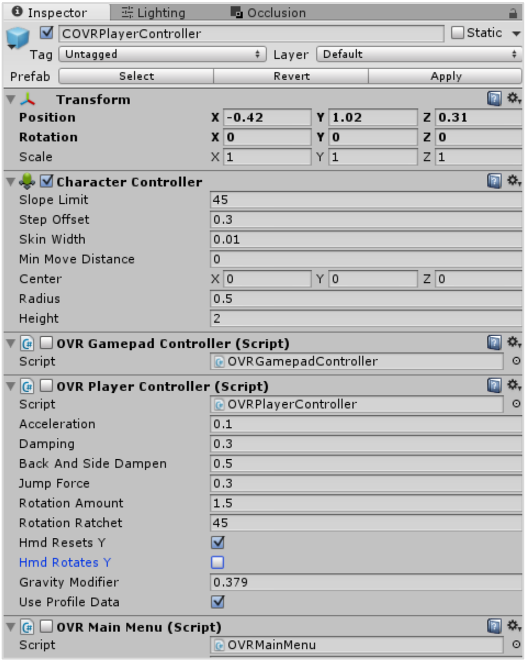
## Setting fully coupled mode on/off

In the COVRPlayerController prefab that is given with our package, the OVRPlayerController script is the one that will control your character, enabling you to use keyboard+mouse input.

By default, "Hmd Rotates Y" should be unchecked.

If you move around, rotating the HMD should just rotate the virtual head, like you would do in real life. To rotate the body, move the mouse or use the right stick of your gamepad.

If you check it, you will see now that rotating the Oculus Rift doesn't rotate just the head but also the body.



☑ COVRPlayerController □ Static

Tag Untagged Layer Default

Prefab

**Transform**

| | | | |
|---|---|---|---|
| Position | X -0.42 | Y 1.02 | Z 0.31 |
| Rotation | X 0 | Y 0 | Z 0 |
| Scale | X 1 | Y 1 | Z 1 |

☑ **Character Controller**

| | |
|---|---|
| Slope Limit | 45 |
| Step Offset | 0.3 |
| Skin Width | 0.01 |
| Min Move Distance | 0 |
| Center | X 0 Y 0 Z 0 |
| Radius | 0.5 |
| Height | 2 |

□ **OVR Gamepad Controller (Script)**

| | |
|---|---|
| Script | OVRGamepadController |

□ **OVR Player Controller (Script)**

| | |
|---|---|
| Script | OVRPlayerController |
| Acceleration | 0.1 |
| Damping | 0.3 |
| Back And Side Dampen | 0.5 |
| Jump Force | 0.3 |
| Rotation Amount | 1.5 |
| Rotation Ratchet | 45 |
| Hmd Resets Y | ☑ |
| Hmd Rotates Y | □ |
| Gravity Modifier | 0.379 |
| Use Profile Data | ☑ |

□ **OVR Main Menu (Script)**

| | |
|---|---|
| Script | OVRMainMenu |

# SDK Documentation

In the CVirtPlayerController script, you'll find an example on how to implement the SDK and use all the Virtualizer capacities.

Here's the description of them, written in the CVirtDevice script.

## GetPlayerHeight

*Returns: float.*
Gives back the current height of the ring construction in centimeters.
0 is default height (upon reset), <0 is when you're getting lower (crouching) and >0 is upper (on your toes, jumping).

## ResetPlayerHeight

*Returns: void.*
Resets the default height of the ring construction.
Basically, you will call this function when pressing a "reset" button for example.

## GetPlayerOrientation

*Returns: Vector3.*
Gives back the orientation as being the forward vector of the body.

Exemple of integration

✓

```
Quaternion rotation = new Quaternion();
rotation.SetLookRotation(virtOrientation, Vector3.up);
transform.localRotation = rotation;
```

## GetMovementSpeed

*Returns: float.*
Gives back the speed of the player's movement in meter per second.

## GetMovementDirection

*Returns: Vector3.*
Gives back the Movement Direction as a normalized vector of structure: (right, 0, forward).

## ResetPlayerOrientation

*Returns: void.*
.Current disabled.