# InstantVR Free Support

Version: 3.8
Date: 19 May, 2017

# Features

The Free version of InstantVR supports the following input:

- keyboard/mouse: using standard WASD input and mouse look functions
- game controllers: support for the followin game controllers is builtin: Xbox, PS4, Steelseries Xl, GameSmart and Sweex GA100 (yes it is obscure, but I happen to have one )
- Oculus Rift DK2, CV1
- HTC Vive without SteamVR controllers (SteamVR controllers require InstantVR Advanced or Edge)
- Samsung Gear VR
- Google Cardboard for Android

# Prerequisites

- The current distribution only supports development on Windows and Android.
- Unity 5.5.0f3 or higher (Unity 5.6 or higher is required for Google Cardboard)

# Getting started

To use InstantVR you need an environment first. The minimum is a flat terrain with a directional light, but you can make it as complex as you want with lots of meshes, rigidbodies and colliders. You are only bound by the limits of Unity and the computer used to drive the game.

You should not include a camera of other first or third person objects in your game. This is fully handled by InstantVR.

To complete the scene you should drag one of the prefabs named ‘MH_…’ from the folder InstantVR into your scene.

The MH_VR prefabs support Oculus Rift, HTC Vive, Gear VR and Cardboard (on Unity 5.6).

The MH_Basic does not include support for any tracking input.

## Video

An introductional tutorial can be found here:
https://www.youtube.com/watch?v=ObtPdsUx9sg

# General configuration

## InstantVR

The instantVR contains references to the 6 transforms which move the avatar. These transforms can be placed anywhere within the Hierarchy.

## IVR_Body Movements

This script translates the movements of the targets to the actual body positions.

It has the following options:

- Enable torso: enables body movements of the upper body
- Enable left: enables leg movements

# Extensions

A number of extensions is included which can be added to the gameObject with InstantVR which add support for one or more input devices or other target controllers.

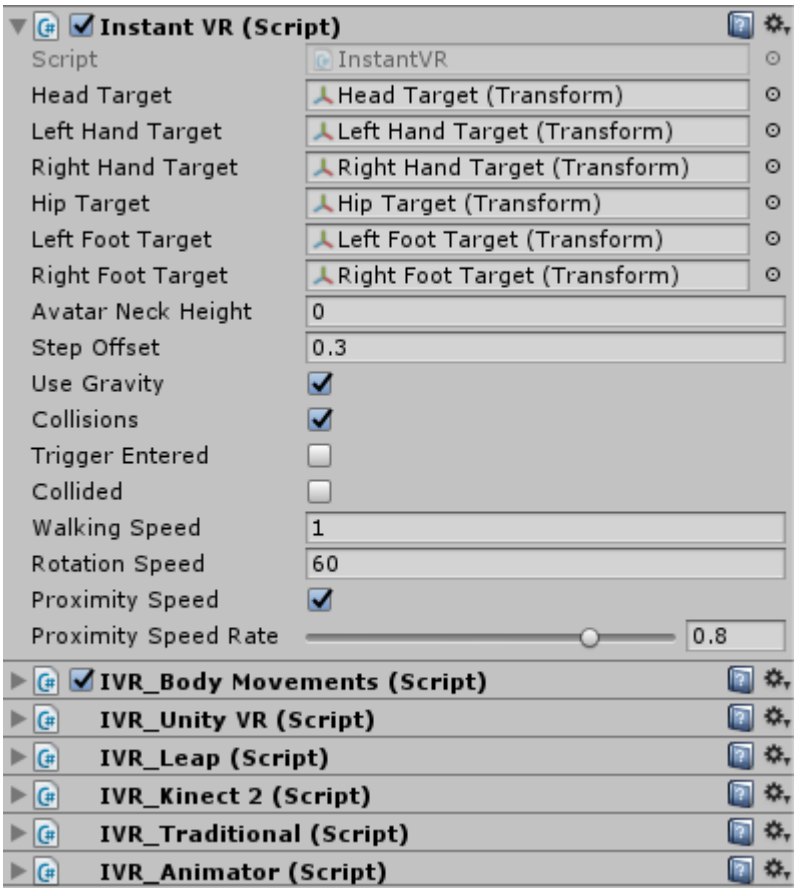
Extensions can be added to every GameObject with an InstantVR script attached to extend the tracking functionality. Extensions will typically implement support for specific input devices like the Oculus Rift or Microsoft Kinect, but also include extensions for animations or networking.

The free version supports the Oculus Rift, Traditional and Animator extensions. More extensions are available in the Advanced edition of InstantVR. These include the HTC Vive, Razer Hydra, Microsoft Kinect 1 & 2 and Leap Motion.

Multiple extensions can be added to a single InstantVR gameObject. The extensions which will be used at play time depends on:

- availability: is the hardware associated with the extension present?
- priority: the extensions with the highest priority are chosed over lower priority extensions.
- tracking: is the hardware currently tracking?

In the example below, we have added 5 extensions: UnityVR (e.g. HTC Vive or Oculus Rift), Leap Motion, Kinect 2, Traditional (Game controller/mouse/keyboard) and Animator.



# Dynamic behaviour

During gameplay with all supported hardware available the behaviour is as follows:

- Oculus Rift will always be used for head tracking. It has the highest priority and is always tracking.
- Leap Motion is used for tracking the hands when the hands are in the field of view of the Leap Motion camers. When the hands are outside the view they cannot be tracked by the Leap Motion so it will not be used then. Hand tracking will drop down to the next hand tracker in the hierarchy
- Kinect supports tracking of all 6 tracking targets, but in this case it will not be used for head tracking, because the Oculus Rift has a higher priority. It will be used for hand tracking when the hands are outside the view of the Leap Motion camera. All other body parts are tracked by Kinect all the time.
- Traditional input is can be used for walking around
- The animator is only used with target are outside the tracking area of the Kinect.

When the game is played with just the Oculus Rift available, it will behave like this:

- Oculus Rift will always be used for head tracking. It has the highest priority and is always tracking.
- Leap Motion and Kinect are not available, so will not be used
- Traditional input is can be used for walking around
- The animator is used to move all targets except the head, which is tracked by the Oculus Rift. This results in leg and arm movements during walking and rotation movements.

Traditional and Animator are typically used when no or limited VR hardware are available and it is good practice to have them as the lowest 2 priority spots as it is shown here.

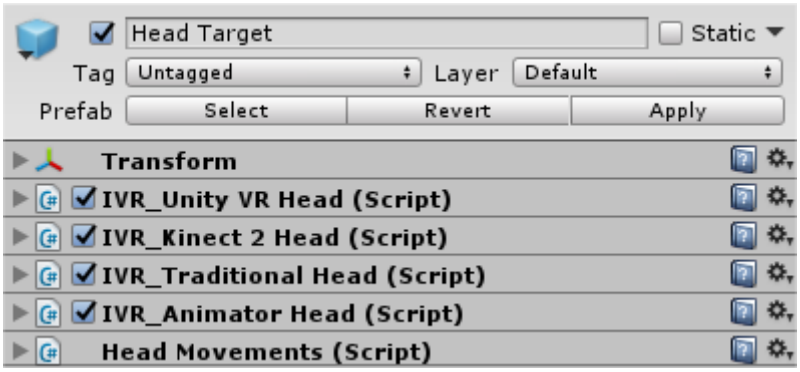
During play mode in the editor it is possible to view which extensions are currently used for each target by switching to debug view:


In this case we have the extensions for Oculus Rift, Razer Hydra, Microsoft Kinect, the traditional and animator extensions configured. In the debug view of InstantVR you can see the various controllers used: Rift for the Head Target, Hydra for the Hand Targets and Kinect for the Hip and Feet Targets.

## Controllers

Every extension implements one or more controllers for targets. The UnityVR extension has one controller: the UnityVR Head Controller which is used solely for the head target. The Leap Motion and Razer Hydra extensions just have a Hand Controller which is used for the left and right hands, while the Kinect extensions have controllers for every target as they can track the full body.

If we look into the Targets themselves we can see the controllers currently associated with that target. Here we have an example of the Head Target:




You can see the UnityVR and Kinect Head Controllers. Like the priority for extensions the order in which the controllers appear determines the priority. This order is in fact controlled by the ordering of the extensions. So you need to reorder the extension when you want to

change the priority of the Head Controllers. Direct changes in the order of controllers will be undone automatically.

*Note: the controller priorities and positions are not updated when the inspector is in Debug mode. This is a known issues and is reported to Unity. The priorities and ordering will be corrected when the inspector is switched back to Normal mode.*

It is possible to disable specific controllers. For instance: if you do not want to use Kinect Head tracking when the Rift is not available, but you do want to use Kinect for the rest of the body, you can disable the Kinect Head Controller. This is done by disabling the IVR_Kinect Head by unchecking the script in the example above. If you want to disable all controllers of an extension, it is better to remove the extension altogether.

## Calibration

Certain extensions need calibration to work correctly. An examples is the Oculus Rift.

The calibration information can be configured in the extensions, but it is also possible to calibrate during gameplay.

Calibration configuration is done by setting the Tracker Position in the extensions. The tracker position is the position of the tracker relative to the player's position. For instance if the tracker is 1 meter in front of you at a height of 1.8 meter from the ground, 20cm to the left, the Tracker Position should be set to x = 0, y = 1.8, z = -0.2

The IR camera of the Oculus Rift can have its tracker positions configured.

The calibration can also be done manually during gameplay using the calibration keys implemented the various input controllers:

Pressing Tab on the keyboard

Pressing Back and Start simultaneously on an Xbox controller

# IVR_UnityVR (Rift, Vive, Gear VR, Cardboard)

Adds support for the Oculus Rift DK2/CV1 or HTC Vive on Windows and Gear VR or Cardboard on Android. Cardboard can also work on iOS, but this is not actively supported.

Important: the option 'Virtual Reality Supported' needs to be checked in the Edit menu, Project Settings, Player.

It is possible to support HTC Vive and Oculus Rift with the same binary by adding both OpenVR and Oculus to the Virtual Reality SDKS in the Player Settings. Just make sure that Oculus is put above the OpenVR SDK.

**Head Target**
Rotational tracking is supported for the Oculus Rift, HTC Vive, Gear VR and Cardboard. Positional tracking is only supported for the Oculus Rift DK2/CV1 and HTC Vive. Note that positional tracking is only possible when the headset is in range of the Oculus IR camera or Lighthouses.

# IVR_Traditional (Mouse, Keyboard, Game Controllers)

Adds support for game controllers and mouse/keyboard.

To support Game controllers, you need to update the InputManager Settings. The package does contain an archive called ‘GameControllerInputSettings.zip’ which contains universal InputManager settings for a the game controllers. Move this to the ProjectSettings folder to get maximum support for your controller.

**Game Controller Input**
InstantVR natively supports a growing number of game controllers. The following controllers are currently supported:

- Microsoft Xbox controller (360, One)
- Playstation4 controller
- Steelseries XL controller
- GameSmart controllers (e.g. MadCatz C.T.R.L.R.)
- Sweex GA100

**Controller Input Sides**
Controllers are split in a left and right side which support the same buttons. On each side the following buttons are supported:

- Thumbstick horizontal and vertical (float values)
- Thumbstick button press (boolean)
- Up, down, left & right (boolean)
- Buttons 0..3 (boolean)
- Bumper (float value)
- Trigger (float value)
- Option (boolean)

For each game controller most buttons can be mapped to these buttons.

**Multiple Controllers**
Currently, the game controller input is limited to one game controller. All available game controllers will be mapped to same input.

**Scripting**
The game controller input can be retrieved using:

```
Controllers.GetController(0)
```

# IVR_Animator

The animator implements procedural animations for legs and arms. It is typically used as the last extension in the list of extensions as a fallback when these are not driven by a input device

## Hand Targets

Follow head: when enabled, the hands will follow the X/Z position of the hips. Needs to be enabled for physical walking using the Rift and when using the IVR_Walking script

Arm swing: enables the arm swing animationg while walking

## Hip Target

Follow head: when enabled, the hip will follow the X/Z position of the head. Needs to be enabled for physical walking using the Rift.

Rotation method determins how the rotation along the Y axis is determined:

- NoRotation: speaks for itself
- LookRotation: the body rotates to the direction in which the player is looking. Good option when using the Oculus Rift.
- HandOrientation: is not supported in the free version. It will behave the same as LookRotation
- Auto: chooses HandOrientation when Hydra is used and LookRotation otherwise

# Walking

Physical walking is supported in combination with the Oculus Rift DK2/CV1, HTC Vive.

# Known issues and limitations

The following issues are known to the current version of InstantVR:

In InstantVR v3, the priority of the controllers is not updated from the extensions priority when the inspector is in Debug Mode. The ordering will be corrected again when the inspector is switched back to Normal mode.

Leg walking animation does not work when Kinect is selected for the foot targets

# Version history

Version 1.0

- Initial release

Version 1.1

- Unity Free support
- Objects can now be throwed
- Included options to enable/disable controllers
- Look rotation using Rift is now working properly
- Leg orientation improved with high foot positions

- Free and Advanced code bases merged

Version 2.0

- Physical drift correction
- Free walking support
- Side stepping
- Removed Character Motor and Character Controller

Version 2.1

- Proximity based walking speed
- Improved leg animation with thumbstick walking
- Improved Rift calibration. No need to face the Rift camera anymore.

Version 3.0

- Unity 5/5.1 support
- Modular architecture
- Body movements in editor
- Realtime switching between input controllers

Version 3.1

- Unity 5.1 support
- Google Cardboard support

Version 3.2

- Gear VR support
- Redesigned calibration

Version 3.4

- Builtin game controller support

Version 3.5

- Native HTC Vive/SteamVR support (unity 5.4)

Version 3.7

- Interaction module with Event System support

Version 3.8

- Native Cardboard support (Unity 5.6)

# More information

The lastest and detailed support information can be found at the Passer VR website: http://passervr.com/support/instantvr-support/

# Contact

You can contact me using the contact form on my website: http://passervr.com/contact/

*Thank you for supporting my work!*
*Pascal.*