

Monado Based OpenXR Virtual Devices

Linux Guide

By Andrew Hazelden <andrew@andrewhazelden.com>

Created 2023-07-23 Updated 2023-08-29 03.36 PM (UTC -4)



The open-source [Monado](#) framework makes it possible to display OpenXR rendered real-time stereoscopic 3D content on a passive stereoscopic 3D monitor like the Schneider Digital [PluraView3D display](#). Monado is cross-platform compatible and works on both Windows and Linux systems.

Monado Documentation

How to use the remote driver

<https://monado.pages.freedesktop.org/monado/howto-remote-driver.html>

Getting started with Monado

<https://monado.freedesktop.org/getting-started.html>

Developing with Monado

<https://monado.freedesktop.org/developing-with-monado.html>

Monado OpenXR Resources

<https://monado.freedesktop.org/openxr-resources.html>

Monado Developer Docs:

<https://monado.pages.freedesktop.org/monado/>

Writing a new OpenXR based Monado HMD Driver

<https://gitlab.freedesktop.org/monado/monado/-/blob/main/doc/writing-driver.md>

Monado Sample Driver

https://monado.pages.freedesktop.org/monado/group__drv__sample.html

Vulkan Extensions

<https://gitlab.freedesktop.org/monado/monado/-/blob/main/doc/vulkan-extensions.md>

OpenComposite

An OpenVR replacement API that forwards calls to OpenXR.

<https://gitlab.com/znixian/OpenOVR>

Generalised Perspective Projection

Perspective Displays and Frustums

<https://web.archive.org/web/20170722004237/http://csc.lsu.edu/~kooima/pdfs/gen-perspective.pdf>

Compiling Monado on Linux

Note: The PluraView3D monitor might have a detectable USB ID or monitor EDID code for the display. This could allow the OpenXR xrt_auto_prober to detect the monitor presence using the vendor ID.

Note: On Linux the "Display Settings" list a PluraView3D display's panel as a "Liyama North America 28" monitor.

Add the Linux build tools

```
sudo apt install g++ gcc cmake git git-lfs ninja-build libsdl2-dev  
zlib1g-dev libcurl-dev
```

Add the supporting libraries

```
sudo apt install ffmpeg libavcodec58 libavcodec-dev libeigen3-dev  
libvulkan-dev libvulkan1 mesa-vulkan-drivers libglx-dev  
libglx-mesa0 libegl-dev libegl-mesa0 libegl1 libegl1-mesa-dev  
libglx0 libegl1-mesa libglu1-mesa-dev mesa-utils libopenvr-api1  
libopenvr-dev libopengl-dev libopenhmd0 libopenhmd-dev  
libopenxr-loader1 libopenxr-dev libopenxr1-monado glslang-tools  
glslang-dev graphviz libann0 libcdt5 libcgmath6 libgl1-mesa-dev  
libglfw3 libglfw3-dev libglm-dev libgts-0.7-5 libgts-bin libgvc6  
libgvpr2 liblab-gamut1 libpathplan4 meson libusb-dev libusb-1.0-0  
libusb-1.0-0-dev libudev-dev libv4l-dev libx11-dev libx11-xcb-dev  
libxrandr-dev libhidapi-libusb0 libhidapi-dev libopencv-core4.5d  
libopencv-core-dev libopencv-dev libuvc0 libuvc-dev libjpeg-dev  
libbluetooth-dev doxygen libgstreamer1.0-0 libgstreamer1.0-dev  
gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps  
libonnx-dev libonnx1 libglew2.2 libglew-dev glew-utils  
libxcompositel1 libxcomposite-dev libxcb-randr0 libxcb-randr0-dev
```

Compile Monado

```
{  
cd $HOME  
git clone https://gitlab.freedesktop.org/monado/monado.git  
cd $HOME/monado  
mkdir build  
cd build  
cmake .. -DCMAKE_BUILD_TYPE=Debug -G "Unix Makefiles"  
cmake --build .  
sudo cmake --build . --target install  
}
```

The Monado command-line executable programs are:

```
/usr/local/bin/monado-cli  
/usr/local/bin/monado-gui  
/usr/local/bin/monado-ctl  
/usr/local/bin/monado-service
```

The OpenXR JSON based configuration file is located at:

```
/usr/local/share/openxr/1/openxr_monado.json
```

Edit the OpenXR JSON config file in the Ubuntu text editor:

```
xed /usr/local/share/openxr/1/openxr_monado.json
```

Start Monado Service

```
export P_OVERRIDE_ACTIVE_CONFIG="remote"  
/usr/local/bin/monado-service
```

Start Monado in the remote mode

```
export P_OVERRIDE_ACTIVE_CONFIG="remote"  
/usr/local/bin/monado-gui remote
```

Start Monado in the simulated mode

```
export P_OVERRIDE_ACTIVE_CONFIG="simulated"  
/usr/local/bin/monado-gui simulated
```

Start Monado in the QWERTY input mode

```
export P_OVERRIDE_ACTIVE_CONFIG="qwerty"  
export QWERTY_ENABLE=1  
export XRT_DEBUG_GUI=1  
/usr/local/bin/monado-service
```

OpenXR Simple Playground

<https://gitlab.freedesktop.org/monado/demos/openxr-simple-playground>

```
{  
cd $HOME  
git clone  
https://gitlab.freedesktop.org/monado/demos/openxr-simple-playgrou  
nd.git  
cd $HOME/openxr-simple-playground/  
cmake -GNinja -Bbuild -DCMAKE_BUILD_TYPE=Release  
ninja -C build  
cd $HOME/openxr-simple-playground/build  
./openxr-playground  
}
```

XR Gears Example

<https://gitlab.freedesktop.org/monado/demos/xrgears>

```
{
cd $HOME
git clone https://gitlab.freedesktop.org/monado/demos/xrgears.git
cd $HOME/xrgears
meson build
ninja -C build
cd $HOME/xrgears/build/src/
./xrgears
}
```

OpenXR Simple Example:

<https://gitlab.freedesktop.org/monado/demos/openxr-simple-example>

```
{
cd $HOME
git clone
https://gitlab.freedesktop.org/monado/demos/openxr-simple-example.
git
cd $HOME/openxr-simple-example/
cmake -GNinja -Bbuild -DCMAKE_BUILD_TYPE=Release
ninja -C build
cd $HOME/openxr-simple-example/build
./openxr-example
}
```

Check the Display Settings on Linux

xrandr is the official configuration utility to the RandR (Resize and Rotate) X Window System extension.

<https://wiki.archlinux.org/title/xrandr>

You can use either of these two commands to get a generic display settings output from the xrandr utility:

```
xrandr --prop
```

```
xrandr
```

The resulting terminal output:

```
Screen 0: minimum 320 x 200, current 3440 x 1440, maximum 16384 x 16384
DP-1 disconnected (normal left inverted right x axis y axis)
HDMI-1 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
HDMI-2 connected primary 3440x1440+0+0 (normal left inverted right x axis y axis) 800mm x 335mm
```

3440x1440	59.97*+ 49.99 29.99
2560x1440	59.97
1920x1440	59.97
2560x1080	60.00 59.94 50.00 60.00
1856x1392	59.97
1792x1344	59.97
2048x1152	59.97
1920x1200	59.97
1920x1080	60.00 60.00 50.00 59.94 59.97
1600x1200	59.97
1680x1050	59.97 59.88
1400x1050	59.97
1600x900	60.00 59.97
1280x1024	60.02 59.97
1400x900	59.97
1280x960	59.97
1440x810	59.97
1368x768	59.97
1280x800	59.97 59.91
1280x720	60.00 50.00 59.94 59.97
1024x768	60.00 59.97
960x720	59.97
928x696	59.97
896x672	59.97
1024x576	59.97
960x600	59.97
960x540	59.97
800x600	60.32 59.97
840x525	59.97
864x486	59.97
720x576	50.00
700x525	59.97
800x450	59.97
720x480	60.00 59.94
640x512	59.97
700x450	59.97
640x480	60.00 59.97 59.94
720x405	59.97
684x384	59.97
640x360	59.97
512x384	59.97
512x288	59.97
480x270	59.97
400x300	59.97
432x243	59.97
320x240	59.97
360x202	59.97
320x180	59.97

DP-3 disconnected (normal left inverted right x axis y axis)

As an xrandr usage example, if you connect the two PluraView3D video cables to the lower pair of display port connectors on a NVIDIA RTX 3090 GPU, the following ports are used:

- DP-2 - USED
- DP-3 - USED

Monado Based Simulator OpenXR Display Canvas Resolution

If you want to change the default resolution of a Monado OpenXR HMD screen, you can edit the following lines of code in the file named "r_hmd.c":

```
// Setup info.
struct u_device_simple_info info;
info.display.w_pixels = 3840*2;
info.display.h_pixels = 2160;
```

Environment Variables

If you want to have the Monado environment variables accessible system-wide you can add the following two entries to the end of your "\$HOME/.profile" document:

```
export
XR_RUNTIME_JSON=/usr/local/share/openxr/1/openxr_monado.json
export P_OVERRIDE_ACTIVE_CONFIG=remote
```

Or

```
export
XR_RUNTIME_JSON=/usr/local/share/openxr/1/openxr_monado.json
export P_OVERRIDE_ACTIVE_CONFIG=qwerty
export QWERTY_ENABLE=1
export XRT_DEBUG_GUI=1
```