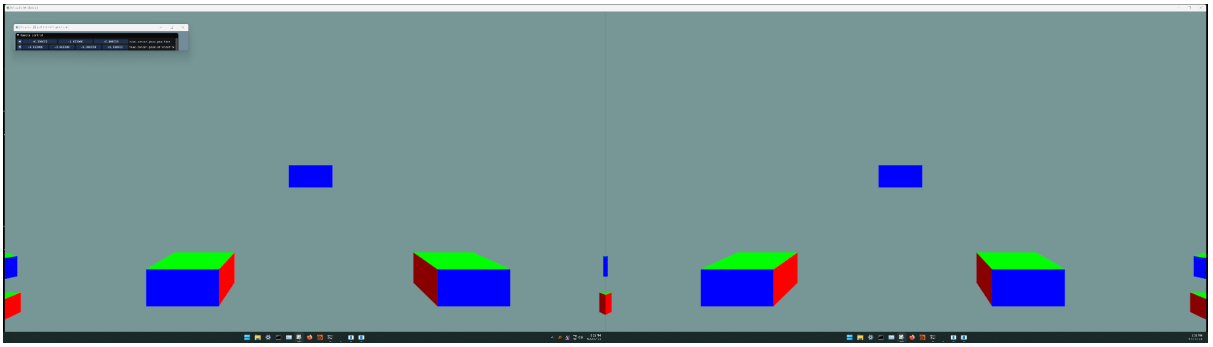
Monado Based OpenXR Virtual Devices

By Andrew Hazelden <andrew@andrewhazelden.com>
Created 2023-07-23 Updated 2023-07-26 08.41 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.

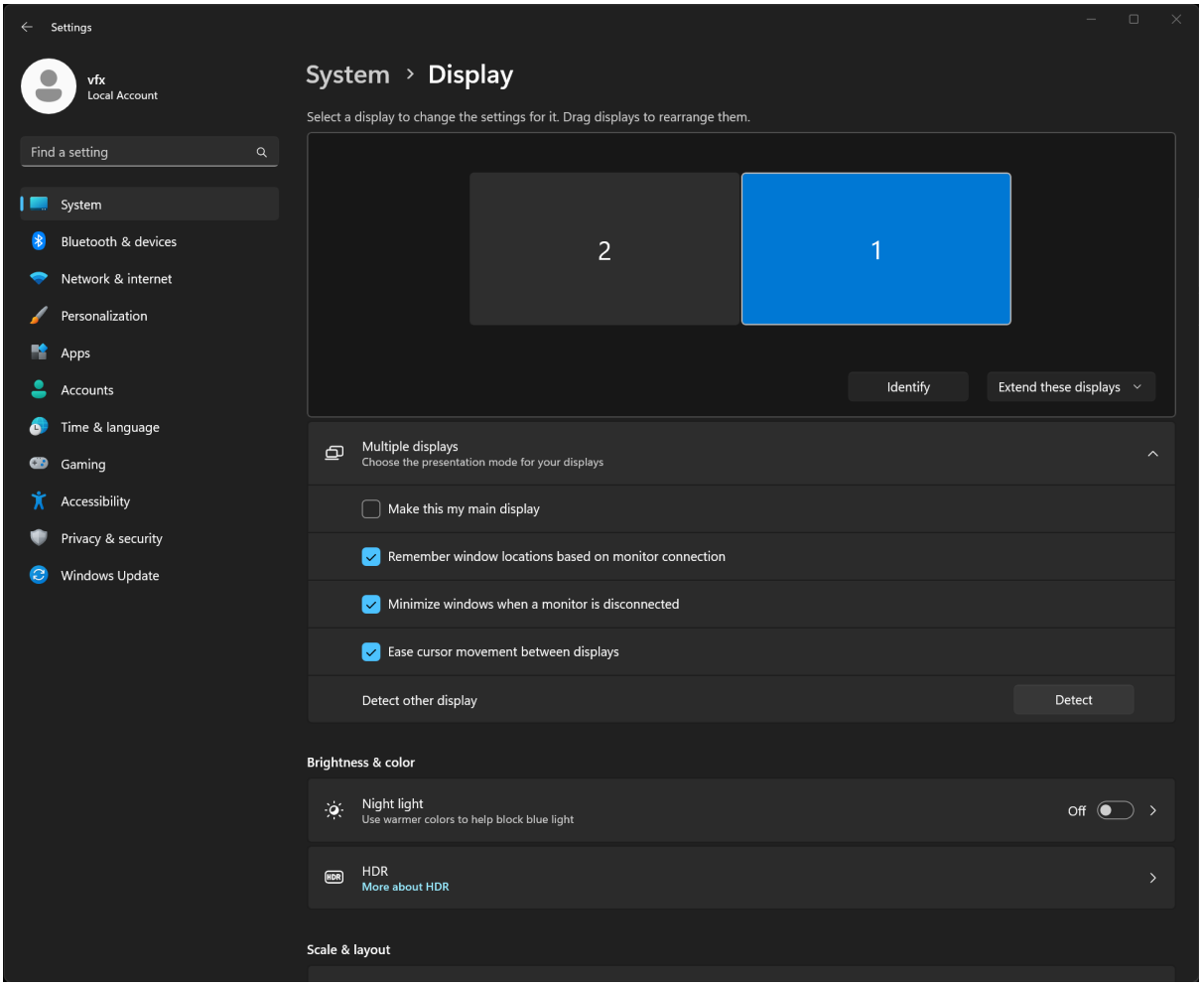
The image below shows the Monado output from the bundled Khronos "Hello XR" program that is included with the Khronos OpenXR-SDK.



The NVIDIA Neural Graphics Primitives NeRF program running with OpenXR output on a PluraView3D monitor is shown below:

At the moment the virtualized OpenXR stereoscopic 3D rendering is achieved using an extended desktop based output mode.

The two PluraView3D display panels need to be arranged in a horizontal layout in the Windows operating system's "System > Display" settings. You can flip the left and right eye ordering by re-positioning display 1 and 2.

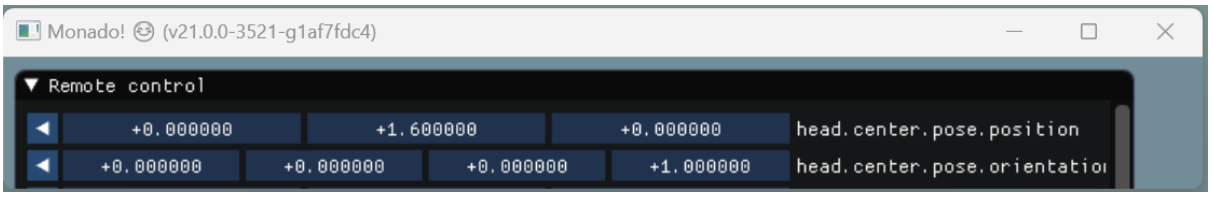


In the future a Vulkan based QuadBuffer stereo output mode would be possible with the addition of a new XRT compositor definition.

Using Monado

Controlling the Camera

For this initial PluraView3D based OpenXR demonstration we are relying on a Monado "remote" OpenXR simulator mode that displays a windowed stereoscopic 3D graphics context. This output is configured through the use of a JSON based configuration file, and environment variables.



The "remote control" user interface has a lot of OpenXR based input settings that can be adjusted. We are primarily focused on the first two controls that adjust the virtual camera's position and orientation:
`head.center.pose.position`
`head.center.pose.orientation`

The position (XYZ translation) and orientation (rotation) controls in the remote control window allow the use of a mouse to mimic the motion of a simulated head mounted display's 6DoF positional tracking sensor.

Tip: When using the Remote Control window, it is helpful to manually resize the bottom-right corner of the window smaller so the view shows only the position and orientation controls. This makes things easier on an extended desktop based monitor setup.

It is possible to define custom tracking devices with Monado. Down the road, hardware like a 3Dconnexion Space Mouse Enterprise could be used to control the OpenXR camera position.

Download the Example

1. A proof-of-concept OpenXR demo is available for download:
http://andrewhazelden.com/projects/kartaverse/downloads/Monado.7z

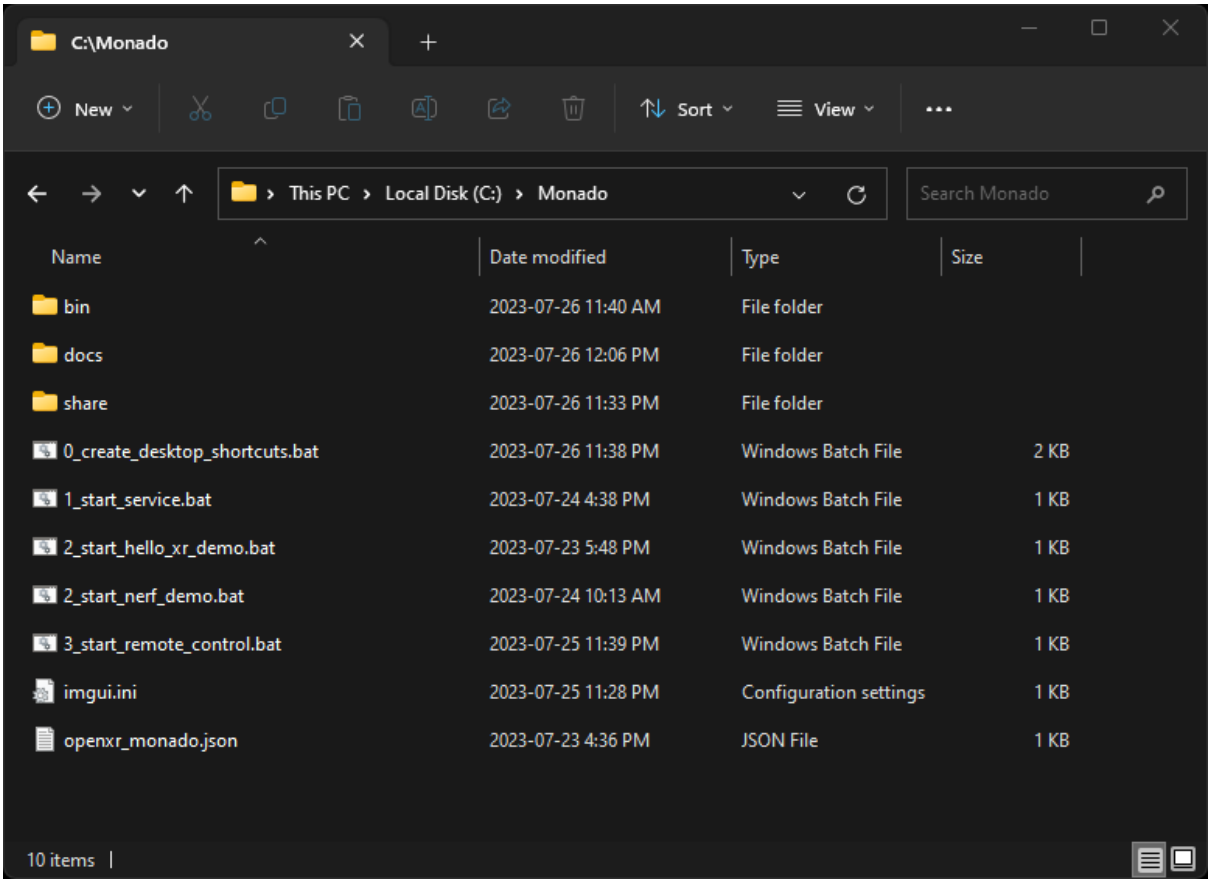
The 7zip file can be expanded to a folder location of your choice on your hard disk. A default location would be to place the expanded folder on disk at:
`C:\Monado\`

2. The NVIDIA InstantNGP NeRF program for Windows can be downloaded from the GitHub page link found here:
https://github.com/NVlabs/instant-ngp#installation

Place the expanded InstantNGP NeRF program on disk at:
`C:\Instant-NGP\`

PluraView3D OpenXR Demo Content

Inside the Monado folder you will see a "bin" folder that holds the Monado libraries, several .bat scripts, and a configuration file in JSON format.

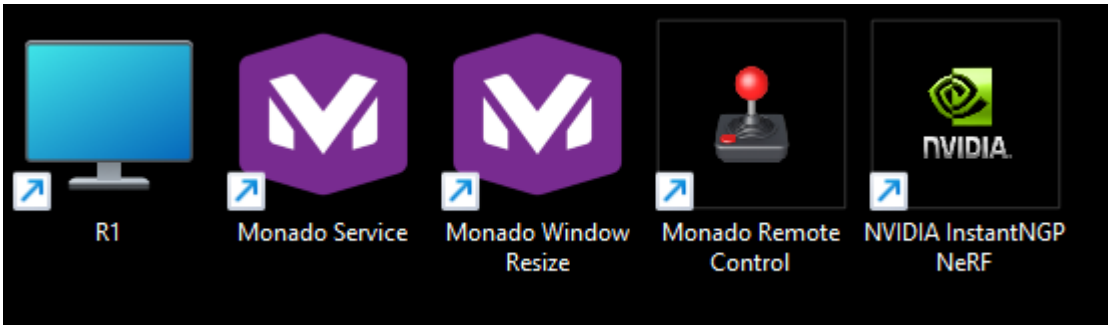


I created several .bat scripts to streamline the process of launching the OpenXR Monado service, the remote control GUI, and a sample Khronos group produced OpenXR program called "Hello XR" that displays several colourful cubes in a stereoscopic 3D environment.

When you want to try out Monado, simply run the scripts in the following order:

- 1_start_service.bat
- 2_start_hello_xr_demo.bat or 2_start_nerf_demo.bat
- 3_start_remote_control.bat

The "0_create_desktop_shortcuts.bat" script creates desktop folder based shortcuts for the included tools.



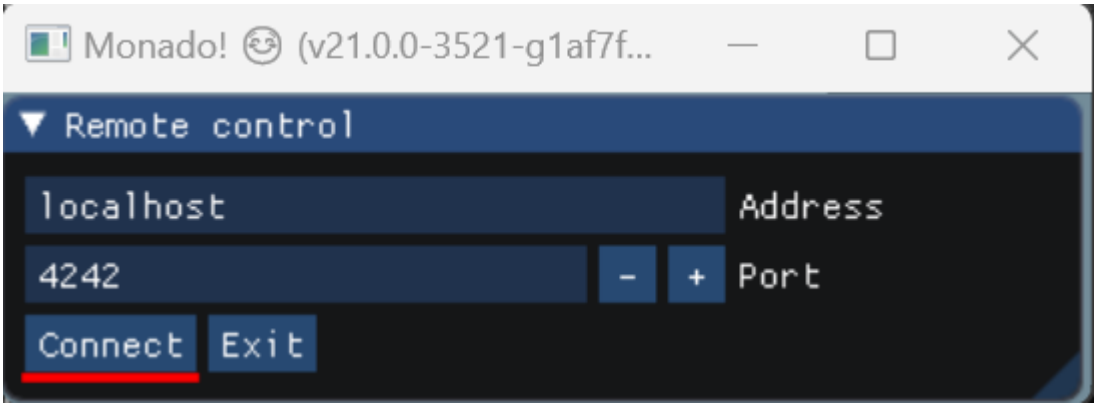
The "1_start_service.bat" script enables the Monado service that registers itself as an OpenXR based virtual HMD known as a "remote" simulator.

The "2_start_hello_xr_demo" script starts a Khronos Group provided sample OpenXR program "Hello XR". The Hello XR demo is set to render the stereoscopic footage using Vulkan graphics. The Monado simulator window is resized to fit the dual monitor layout "extended desktop" screen size.

The "2_start_nerf_demo.bat" script starts the NVIDIA InstantNGP NeRF rendering program in the OpenXR based VR mode. The pre-compiled InstantNGP program needs to be manually downloaded and installed to "C:\Instant-NGP\instant-ngp.exe".

The "3_start_remote_control" script displays a Monado utility window. Click the "Remote" button in this window to display the Remote Control.

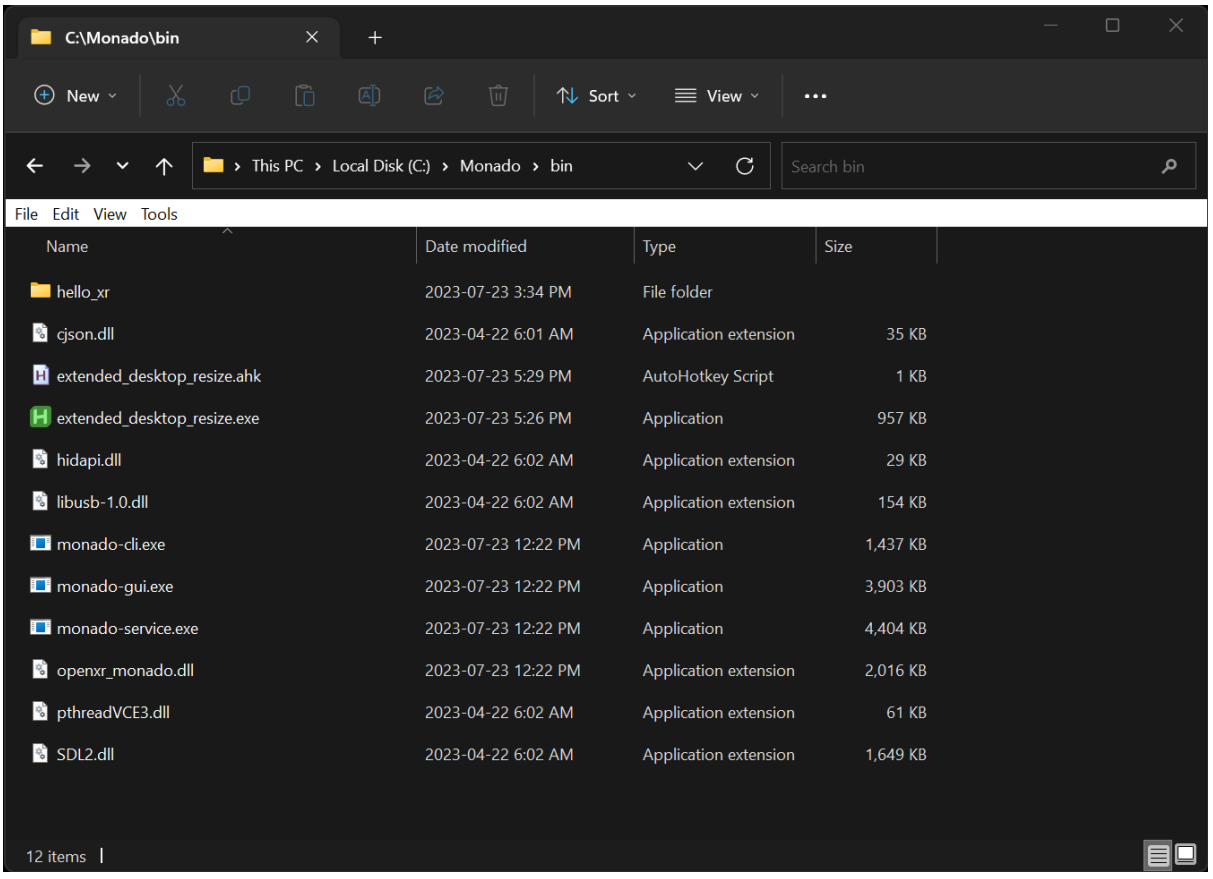
Then press the "Connect" button to connect to the active OpenXR session. You can then move the VR camera view around using your mouse. It is useful to scale the borders of the Remote Control window down in size so it takes up less space on the monitor.



An interesting feature of the Remote Control window is you can use an IP based network computer connection to drive the OpenXR session externally.

Monado Executables

Inside the Monado "bin" folder you will see several executables:



Monado includes three executables called "monado-cli.exe", "monado-gui.exe", and "monado-service.exe". There are several supporting .dll libraries.

Environment Variables

If you want to have the Monado environment variables accessible system-wide you can add the following two entries to your Windows operating system's list of active environment variables:

XR_RUNTIME_JSON=C:\Monado\openxr_monado.json
P_OVERRIDE_ACTIVE_CONFIG=remote

Resizing the Graphics Context

To make it easier to accurately position the Monado graphics context on a dual monitor PluraView3D display, I created a simple script called "extended_desktop_resize". This script was written using a toolset called AutoHotkey. The script checks the combined resolution of the two monitor displays and scales the Monado window to that exact size.

For completeness, here is an inline copy of the "extended_desktop_resize.ahk" script code:

```
#Requires AutoHotkey v2.0
#SingleInstance

; Resize the Monado window to fit across a dual monitor extended
desktop configuration
WinWait("ahk_class Monado")
VirtualScreenWidth := SysGet(78)
VirtualScreenHeight := SysGet(79)
WinMove 0, 0, VirtualScreenWidth, VirtualScreenHeight, "ahk_class
Monado"
```

This script is compiled by AutoHotkey into an executable named "extended_desktop_resize.exe". The executable is run automatically by the "2_start_hello_xr_demo.bat" and " 2_start_nerf_demo.bat" scripts so it will resize the graphics context as soon as the OpenXR window is created.

Bat Scripts

Included below are inline copies of the bat script that were created for this presentation.

"0_create_desktop_shortcuts.bat" Source Code

```
@echo off

REM Create Desktop Shortcuts

REM Change the working directory to the current Monado folder (ex.
C:\Monado\)
cd %~dp0\

REM Create a Monado Service shortcut using a Windows Powershell
command
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive
-NoProfile -Command $ws = New-Object -ComObject WScript.Shell;
$DesktopPath = $ws.SpecialFolders('Desktop') +  '\Monado
Service.lnk'; $s = $ws.CreateShortcut($DesktopPath); $S.TargetPath
= (pwd).Path + '/1_start_service.bat'; $S.IconLocation =
(pwd).Path + '/share/monado_icon.ico'; $S.Save()

REM Create a Monado Window Resize shortcut using a Windows
Powershell command
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive
-NoProfile -Command $ws = New-Object -ComObject WScript.Shell;
$DesktopPath = $ws.SpecialFolders('Desktop') +  '\Monado Window
Resize.lnk'; $s = $ws.CreateShortcut($DesktopPath); $S.TargetPath
=  (pwd).Path + '/bin/extended_desktop_resize.exe';
$S.IconLocation = (pwd).Path + '/share/monado_icon.ico'; $S.Save()

REM Create a Monado Remote Control shortcut using a Windows
Powershell command
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive
-NoProfile -Command $ws = New-Object -ComObject WScript.Shell;
$DesktopPath = $ws.SpecialFolders('Desktop') +  '\Monado Remote
Control.lnk'; $s = $ws.CreateShortcut($DesktopPath); $S.TargetPath
=  (pwd).Path + '/3_start_remote_control.bat'; $S.IconLocation =
(pwd).Path + '/share/joystick_icon.ico'; $S.Save()

REM Create a NVIDIA InstantNGP NeRF shortcut using a Windows
Powershell command
```

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive
-NoProfile -Command $ws = New-Object -ComObject WScript.Shell;
$DesktopPath = $ws.SpecialFolders('Desktop') +  '\NVIDIA
InstantNGP NeRF.lnk'; $s = $ws.CreateShortcut($DesktopPath);
$S.TargetPath =  (pwd).Path + '/2_start_nerf_demo.bat';
$S.IconLocation = (pwd).Path + '/share/nvidia_icon.ico'; $S.Save()
```

"1_start_service.bat" Source Code

```
@echo off

REM Specify the OpenXR configuration file
set XR_RUNTIME_JSON=%~dp0\bin\openxr_monado.json

REM Specify Monado environment variables
set P_OVERRIDE_ACTIVE_CONFIG=remote

REM Launch the Monado OpenXR based background service
%~dp0\bin\monado-service.exe

pause
```

"2_start_hello_xr_demo.bat" Source Code

```
@echo off

REM Specify the OpenXR configuration file
set XR_RUNTIME_JSON=%~dp0\openxr_monado.json

REM Specify Monado environment variables
set P_OVERRIDE_ACTIVE_CONFIG=remote

REM Resize the Monado Window to fit across a dual monitor extended
desktop layout
start "Extended Desktop Resize"
%~dp0\bin\extended_desktop_resize.exe

REM Start the OpenXR Based hello_xr demo app
%~dp0\bin\hello_xr\hello_xr.exe -G Vulkan

pause
```

"2_start_nerf_demo.bat" Source Code

```
@echo off
```

```
REM Specify the OpenXR configuration file
set XR_RUNTIME_JSON=%~dp0\openxr_monado.json

REM Specify Monado environment variables
set P_OVERRIDE_ACTIVE_CONFIG=remote

REM Resize the Monado Window to fit across a dual monitor extended
desktop layout
start "Extended Desktop Resize"
%~dp0\bin\extended_desktop_resize.exe

REM Start the NVIDIA Instant-NGP NeRF App
C:\Instant-NGP\instant-ngp.exe --vr C:\Instant-NGP\data\nerf\fox

pause
```

"3_start_remote_control.bat" Source Code

```
@echo off

REM Specify the OpenXR configuration file
set XR_RUNTIME_JSON=%~dp0\openxr_monado.json

REM Specify Monado environment variables
set P_OVERRIDE_ACTIVE_CONFIG=remote

REM Launch the Mondao Remote Control GUI used to adjust the OpenXR
camera view
%~dp0\bin\monado-gui.exe

pause
```

"Openxr_monado.json" Configuration File

```
{
      "file_format_version": "1.0.0",
      "runtime": {
      "name": "Monado",
      "library_path": "bin\\openxr_monado.dll"
      },
      "active": "remote",
      "remote": {
      "version": 0,
      "port": 4242
      }
}
```

Further Development

The next phase of R&D is focusing on improving the user input control options available in a customized version of the "remote control" interface. I am binding a USB gamepad to the interface.

The final graphics context is displayed using the SDL2 graphics library. Work is being carried out now to make it easier to toggle quickly between a dual-monitor "extended desktop" windowed, or full-screen display mode.

The next documentation effort is to cover how to set up OpenXR to work alongside a web browser's native implementation of WebXR.

Additionally, there is an interesting library called OpenComposite that is able to work alongside Monado. The OpenComposite library allows the OpenXR API to be used to run older OpenVR/WebVR based applications and content. OpenComposite translates each of the OpenVR API function calls into a similar OpenXR based implementation. OpenComposite can be installed at a system-wide level using a "OpenComposite Runtime Switcher" application, or at a per-application level using an updated "openvr_api.dll" library that replaces the default SteamVR library in the individual program's bin folder.