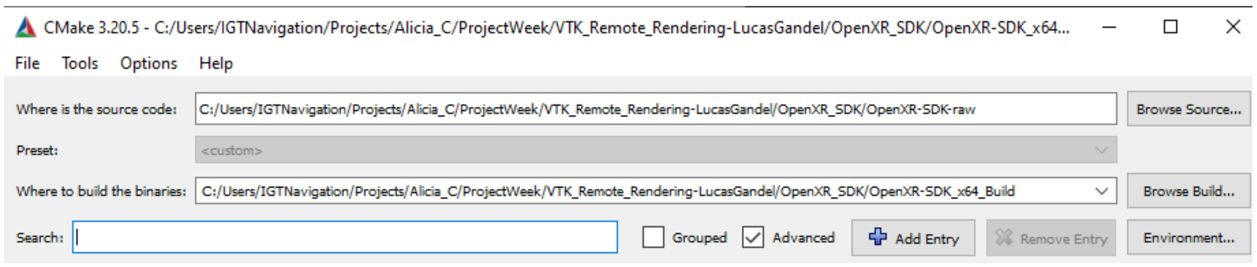
HoloLens 2 support using OpenXr Remoting

This tutorial will allow you to stream directly to your HoloLens 2 using VTK, from scratch. It is a 3-step process. First, you will build and install the OpenXR SDK in your computer. Then, you will build the VTK and finally, you will try a sample.

OpenXR SDK

Build and install the OpenXR_SDK

1. Open CMake and select the OpenXR-SDK(-raw) folder in the source code. Create another folder right next to it called "OpenXR-SDK_x64_Build". Click on "Configure".



CMake 3.20.5 - C:/Users/IGTNavigation/Projects/Alicia_C/ProjectWeek/VTK_Remote_Rendering-LucasGandel/OpenXR_SDK/OpenXR-SDK_x64...

Where is the source code: C:/Users/IGTNavigation/Projects/Alicia_C/ProjectWeek/VTK_Remote_Rendering-LucasGandel/OpenXR_SDK/OpenXR-SDK-raw

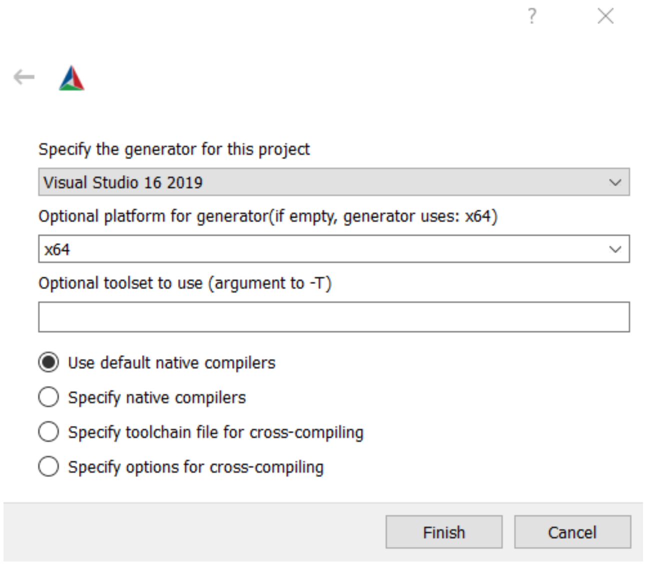
Preset: <custom>

Where to build the binaries: C:/Users/IGTNavigation/Projects/Alicia_C/ProjectWeek/VTK_Remote_Rendering-LucasGandel/OpenXR_SDK/OpenXR-SDK_x64_Build

Search:

☐ Grouped ☑ Advanced

2. If you haven't created the build folder in advance, CMake will ask you to create it automatically. Select "Yes". Then, the following dialog will display. You must specify your Visual Studio Version (For instance, Visual Studio 16 2019) and the platform for the generator (x64).



Specify the generator for this project: Visual Studio 16 2019

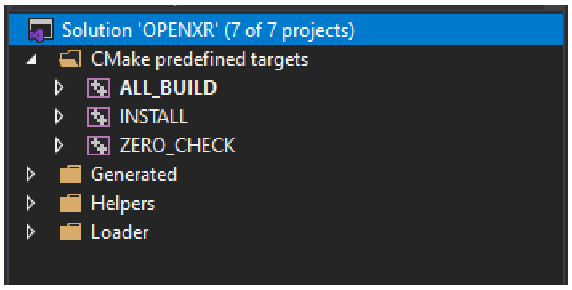
Optional platform for generator(if empty, generator uses: x64): x64

Optional toolset to use (argument to -T):

● Use default native compilers
○ Specify native compilers
○ Specify toolchain file for cross-compiling
○ Specify options for cross-compiling

3. Use the "Search" region to find options with CMake (make sure the "Advanced" checkbox is on) and change the **CMAKE_INSTALL_PREFIX** to the path "[same path as the one of OpenXR-SDK_x64_Build but this time finished in OpenXR-SDK_x64_Install]"

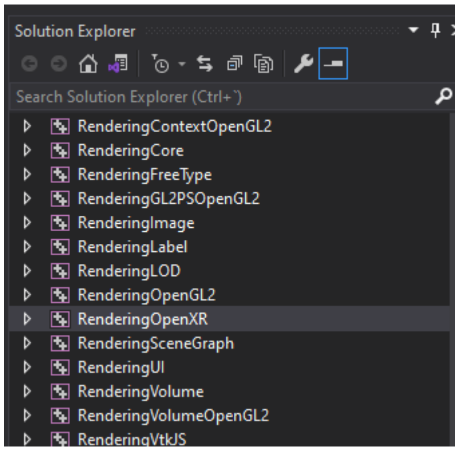
4. Set the **DYNAMIC_LOADER** option to ON.
5. Configure > Generate > Open Project
6. In Visual Studio, make sure you work in **Release x64 mode**. Then, right click on ALL_BUILD inside "CMake predefined targets" and select "**Build**". You can also build everything by right clicking on the main "Solution 'OPENXR'".
7. Once finished, right click on **INSTALL** right below ALL_BUILD and select "Project Only > Build only INSTALL".



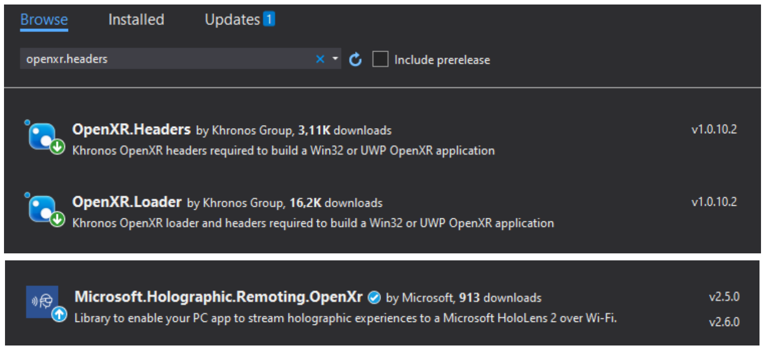
VTK

Build the VTK.

1. Open CMake and select the vtk-OpenXR-DX-Remoting(-raw) folder in the source code. Create another folder right next to it called "vtk-OpenXR-DX-Remoting_x64_Build". If you don't create it in advance, it will be created automatically.
2. Change the **CMAKE_INSTALL_PREFIX** to a path like "[same path as the one of vtk-OpenXR-DX-Remoting_x64_Build but this time finished in vtk-OpenXR-DX-Remoting_x64_Install]"
3. Set the **VTK_MODULE_ENABLE_VTK_RenderingOpenXR** option from DEFAULT to YES.
4. Select "Configure" and "Generate". You will get an error.
5. Change the paths of the variables: **OpenXR_LIBRARY** and **OpenXR_INCLUDE_DIRS** (which would appear as "NOTFOUND") to:
    a. OpenXR_LIBRARY: *C:/Users/IGTNavigation/Projects/Alicia_C/ProjectWeek/VTK_Remote_Rendering-LucasGandel/1-OpenXR_SDK/OpenXR-SDK_x64_Install/lib/openxr_loaderd.lib*
    b. OpenXR_INCLUDE_DIR: *C:\Users\IGTNavigation\Projects\Alicia_C\ProjectWeek\VTK_Remote_Rendering-LucasGandel\1-OpenXR_SDK\OpenXR-SDK_x64_Install\include\openxr*
6. Select "Configure", "Generate" and "Open Project".
7. In visual Studio, make sure you work in **Release x64 mode**. Then, look at the Solution Explorer and scroll down until "**RenderingOpenXR**". Right click on it -> Manage Nuget Packages. Install the following:
    a. **Microsoft.Holographic.Remoting.OpenXr** (version 2.5.0)
    b. **OpenXR.Headers** (default version)
    c. **OpenXR.Loader** (default version)



RenderingContextOpenGL2
RenderingCore
RenderingFreeType
RenderingGL2PSOpenGL2
RenderingImage
RenderingLabel
RenderingLOD
RenderingOpenGL2
RenderingOpenXR
RenderingSceneGraph
RenderingUI
RenderingVolume
RenderingVolumeOpenGL2
RenderingVtkJS

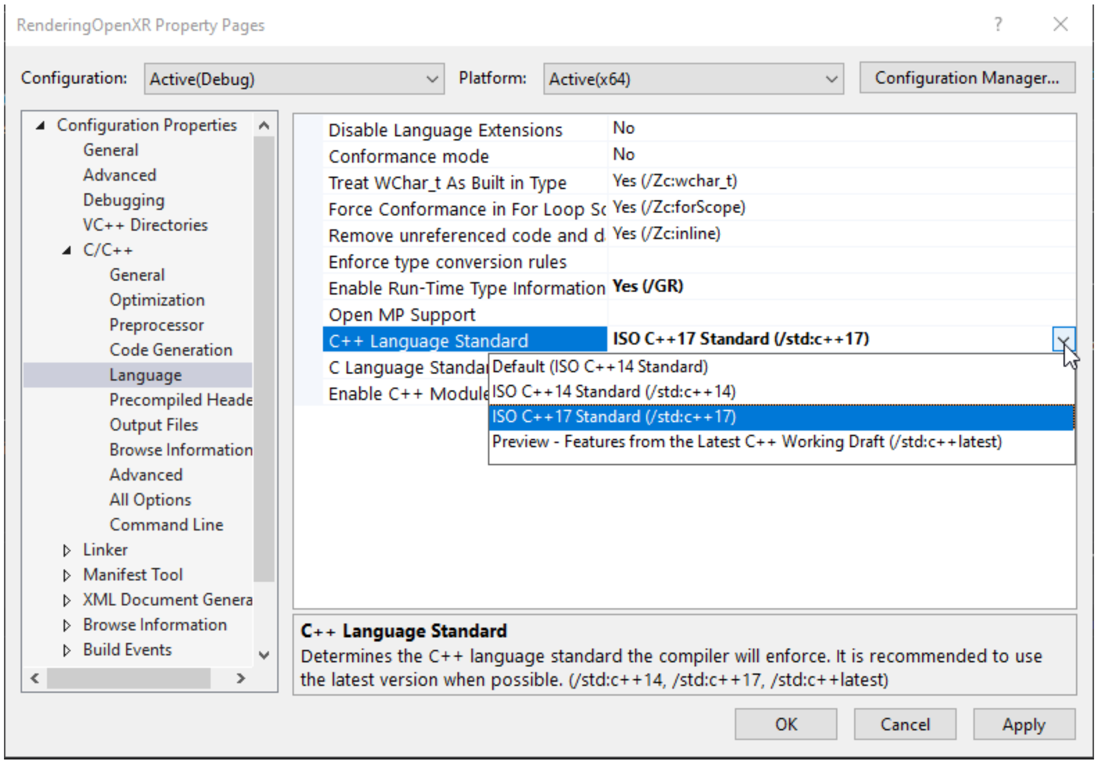


OpenXR.Headers by Khronos Group, **3,11K** downloads — v1.0.10.2
Khronos OpenXR headers required to build a Win32 or UWP OpenXR application

OpenXR.Loader by Khronos Group, **16,2K** downloads — v1.0.10.2
Khronos OpenXR loader and headers required to build a Win32 or UWP OpenXR application

Microsoft.Holographic.Remoting.OpenXr by Microsoft, **913** downloads — v2.5.0 v2.6.0
Library to enable your PC app to stream holographic experiences to a Microsoft HoloLens 2 over Wi-Fi.

8. Right click again on “RenderingOpenXR” and select Properties > C/C++ > Language > C++ Language Standard > ISO C++ 17 Standard (/std:c++17). Then, apply changes and select OK.



Disable Language Extensions	No
Conformance mode	No
Treat WChar_t As Built in Type	Yes (/Zc:wchar_t)
Force Conformance in For Loop Sc	Yes (/Zc:forScope)
Remove unreferenced code and d	Yes (/Zc:inline)
Enforce type conversion rules	
Enable Run-Time Type Information	**Yes (/GR)**
Open MP Support	
C++ Language Standard	**ISO C++17 Standard (/std:c++17)**
C Language Standa	
Enable C++ Module	

Default (ISO C++14 Standard)
ISO C++14 Standard (/std:c++14)
ISO C++17 Standard (/std:c++17)
Preview - Features from the Latest C++ Working Draft (/std:c++latest)

C++ Language Standard
Determines the C++ language standard the compiler will enforce. It is recommended to use the latest version when possible. (/std:c++14, /std:c++17, /std:c++latest)

9. Right click on “Solution ‘VTK’” in the top of the Solution Explorer and select “Build Solution”.

Test

Build the Lucas Gandel's VTK test-dev.

1. Repeat the same steps as for building the VTK with the following configuration in CMake:
   Source code path:
   C:/Users/IGTNavigation/Desktop/Alicia_Desktop/Project_Week/VTKRemoteRendering/3-VTK_Test-dev_LucasExample-raw
   Where to build the binaries:
   C:/Users/IGTNavigation/Desktop/Alicia_Desktop/Project_Week/VTKRemoteRendering/3-VTK_Test-dev_LucasExample_x64_Build
   a. **OpenXR_DIR =** *C:\Users\IGTNavigation\Desktop\Alicia_Desktop\Project_Week\VTKRemoteRendering\1-OpenXR-SDK_x64_Install\cmake*
   b. **VTK_DIR =** *C:\Users\IGTNavigation\Desktop\Alicia_Desktop\Project_Week\VTKRemoteRendering\2-vtk_x64_Build*
   c. **OpenXR_LIBRARY =** *C:\Users\IGTNavigation\Desktop\Alicia_Desktop\Project_Week\VTKRemoteRendering\1-OpenXR-SDK_x64_Install\lib\openxr_loader.lib*
   d. **OpenXR_INCLUDE_DIR =** *C:\Users\IGTNavigation\Desktop\Alicia_Desktop\Project_Week\VTKRemoteRendering\1-OpenXR-SDK_x64_Install\include\openxr*

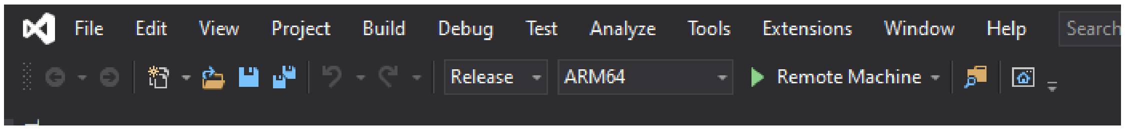
   You may have to configure at some point until all the options are visible.
2. Also, don't forget to change again the **CMAKE_INSTALL_PREFIX** to a path like "[same path as the one of 3-VTK_Test-dev_LucasExample_x64_Build but this time finished in 3-VTK_Test-dev_LucasExample_x64_Install]"
3. Configure > Generate > Open Project
4. In visual studio, set the mode to **Release x64.**
5. Right click on **TestOpenXRInitialization** project in the Solution Explorer and add the same NuGet packages as to the VTK:
   a. **Microsoft.Holographic.Remoting.OpenXr** (version 2.5.0)
   b. **OpenXR**.**Headers** (default version)
   c. **OpenXR**.**Loader** (default version)
6. Right click on "ALL_BUILD" or "Solution 'TestOpenXRInitialization'" and select **Build solution** / Build

Microsoft's Oficial sample

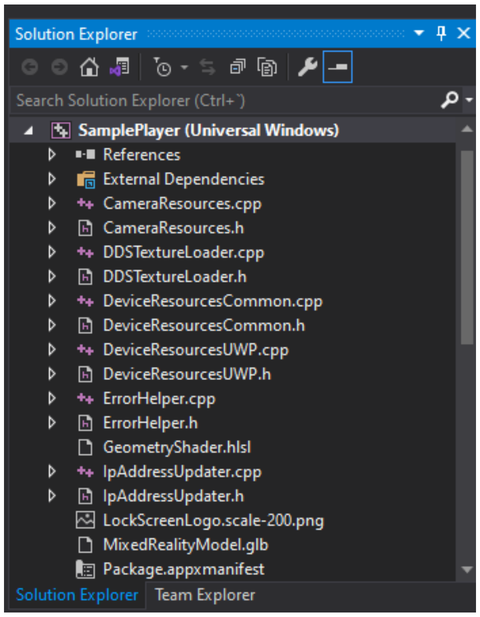
1. Download the official Mixed Reality Holographic Remoting sample from:
   https://github.com/microsoft/MixedReality-HolographicRemoting-Samples/tree/Version_2.5.0
2. Unzip the downloaded folder and travel to
   C:\Users\IGTNavigation\Desktop\Alicia_Desktop\Project_Week\VTKRemoteRendering\4-

MixedReality-HolographicRemoting-Samples-master\player\sample. Once there, double click in the **SamplePlayer.sln** file to open it with Visual Studio.

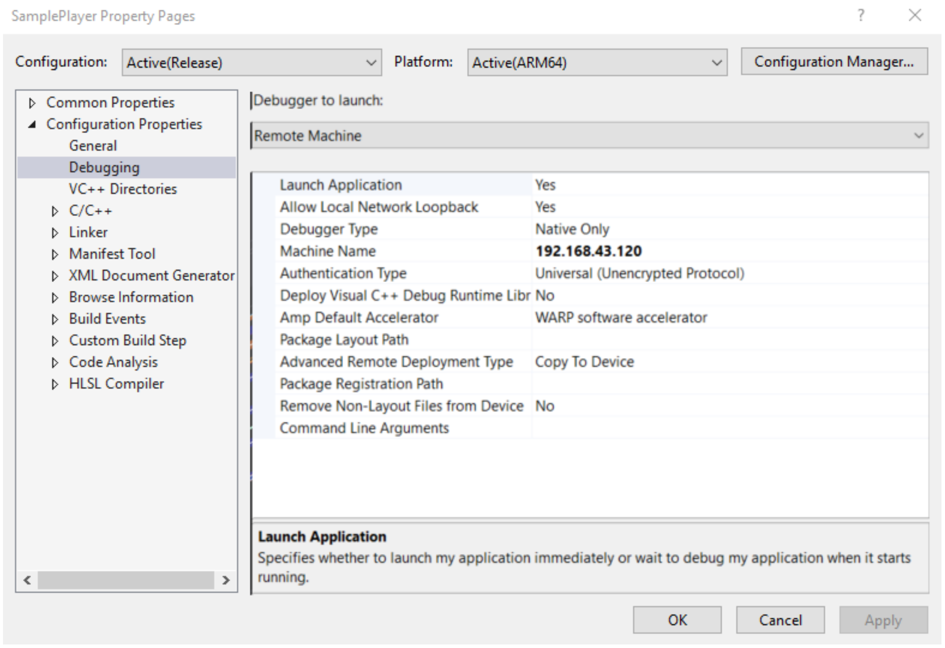
3. Select the mode "**Release ARM64 Remote Machine**"



4. Right click on **SamplePlayer (Universal Windows)** project inside the Solution explorer and select "**Properties**".



5. Configuration Properties > Debugging > Machine Name. Set your **HoloLens 2 IP address.** You can find it in your HoloLens, inside the advanced configuration of your wifi. Alternatively, you can see inside the "HoloRemoting" app of your HoloLens.



Configuration: Active(Release) Platform: Active(ARM64)

Debugger to launch: Remote Machine

Launch Application	Yes
Allow Local Network Loopback	Yes
Debugger Type	Native Only
Machine Name	**192.168.43.120**
Authentication Type	Universal (Unencrypted Protocol)
Deploy Visual C++ Debug Runtime Libr	No
Amp Default Accelerator	WARP software accelerator
Package Layout Path	
Advanced Remote Deployment Type	Copy To Device
Package Registration Path	
Remove Non-Layout Files from Device	No
Command Line Arguments	

Launch Application
Specifies whether to launch my application immediately or wait to debug my application when it starts running.

6. Right click on “Solution ‘SamplePlayer’” and select “**Restore NuGet packages**”
7. Press on “Remote Machine” to execute the solution.
8. Once it is running, you can launch your own .sln file on a parallel Visual Studio window to see your models in your HoloLens.
9. You are all set!

Thank you for reading, we hope this tutorial helped,

Lucas Gandel and Alicia Pose Díez de la Lastra