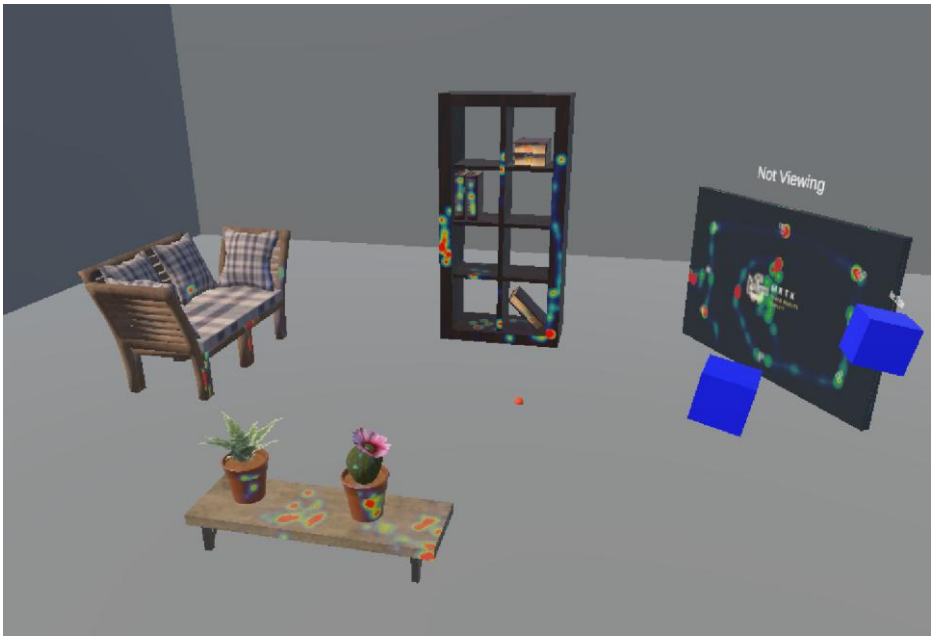
Initial Testing with Unity 2022.3 LTS, MRTK 2.8 & OpenXR 1.11.2 Plugin to Capture & Visualize Gaze Data

Sections

1 Results

By using Unity 2022.3 LTS, MRTK 2.8 & OpenXR 1.11.2 Plugin it is **possible to capture gaze data on 3D game objects in Unity**. Furthermore, **store and visualize gaze data** that was captured. Before proceeding with the rest of this report please view the short demo of eye-tracking (simulation using mouse as gaze data):
https://drive.google.com/file/d/1gRNhuo0GHouzpT--2aifzOa4ZugmQlQy/view?usp=sharing

My Sample Unity Project (GitHub): https://github.com/luhouyang/SampleDazeData.git



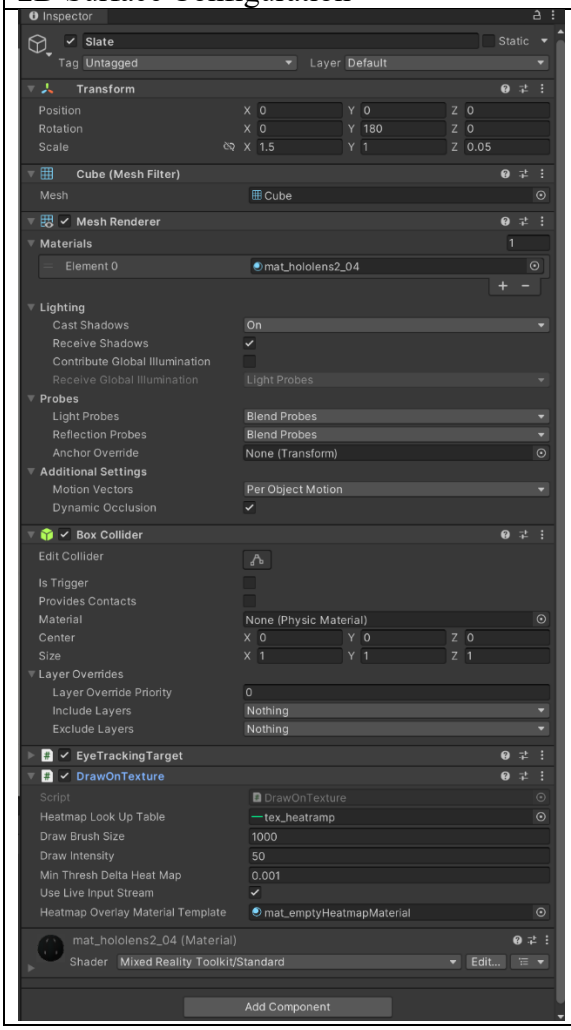
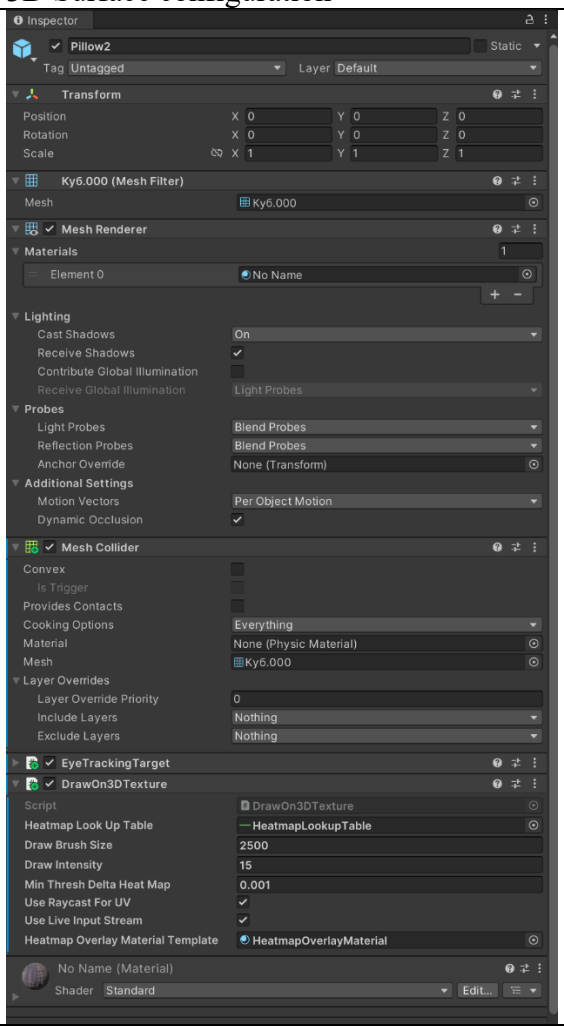
After reading the documentation [9] from Microsoft the code and scene setup **do not require much change** to switch between '**simulation mode**' and '**running on HoloLens2**'. The changes I have found are:

- In the Unity project navigate to **SCENE** > **MixedReality Toolkit** > **Input** > **Input Data Provide**rs, change '**Input Simulation Service**' to '**OpenXR XRSDK Eye Gaze Provider**'. Created according to steps in [9]
- Ensure **GazeInput** is configured according to **steps here**.
- Enable '**Use Eye Tracking Data**' at **SCENE** > **MixedReality Toolkit** > **Input** > **Pointers**.
- Ensure the project was set up correctly according to [3] (same setup for simulation).
- Build and compile the HoloLens 2 MRTK app.
- Complete eye calibration on HoloLens 2.
- Grant permission to use eye tracking when prompted (if no prompt was shown, there is an issue with **GazeInput** configuration).

Another video on the Microsoft Eye-tracking Demo: https://drive.google.com/file/d/1O-mKdVdSKmTLSNI-LQeAYyGjk4bR-wKn/view?usp=sharing

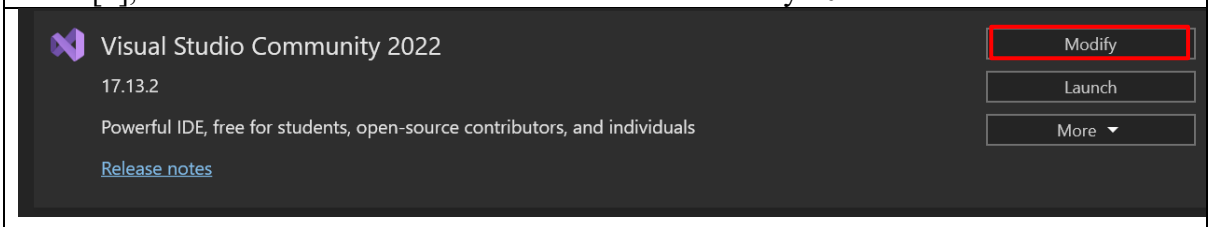
Microsoft Eye-tracking Demo (GitHub): https://github.com/microsoft/MixedRealityToolkit-Unity/tree/main/Assets/MRTK/Examples/Demos/EyeTracking

HoloLens 2 Emulator (not yet explored, requires BIOS setting change and Hyper-V): https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-hololens-emulator

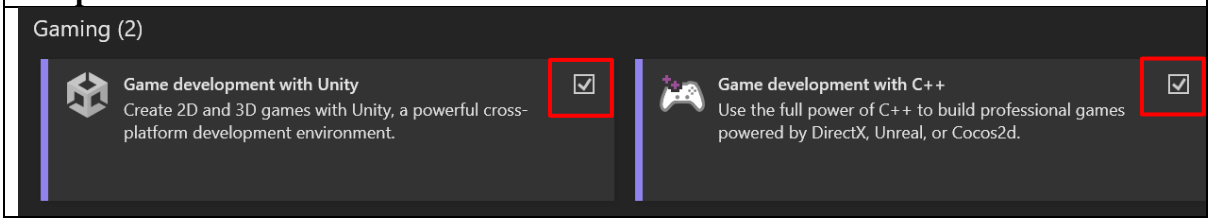
2D Surface Configuration	3D Surface configuration
	

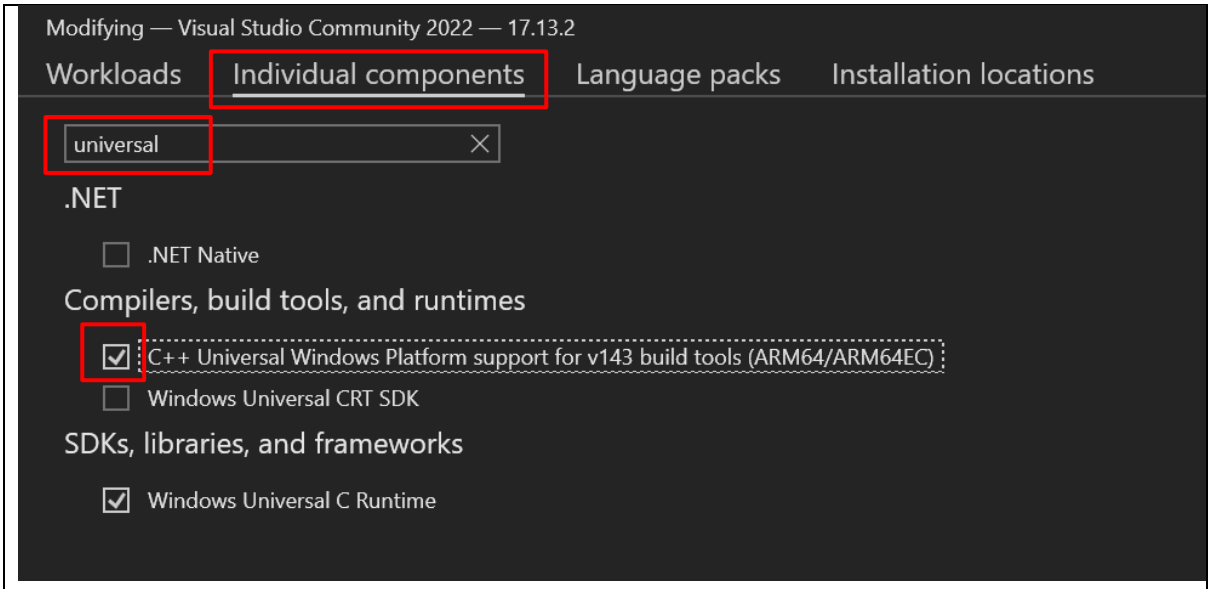
2 Unity 2022.3 LTS, MRTK 2.8 & OpenXR 1.11.2 setup

Read [2], Check UWP Modules & Visual Studio Community 2022 Version

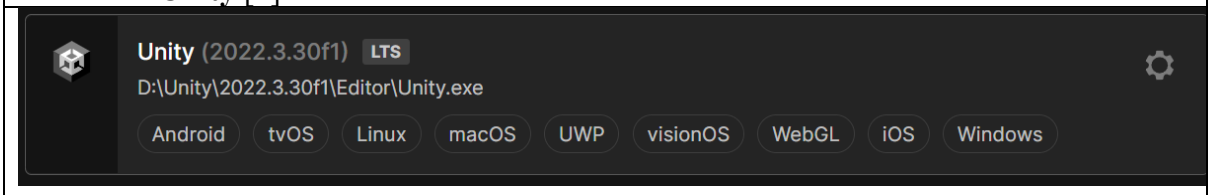


Modify the installation by adding **Game development options** & by going to '**Individual components**' and search '**universal**' add **UWP**

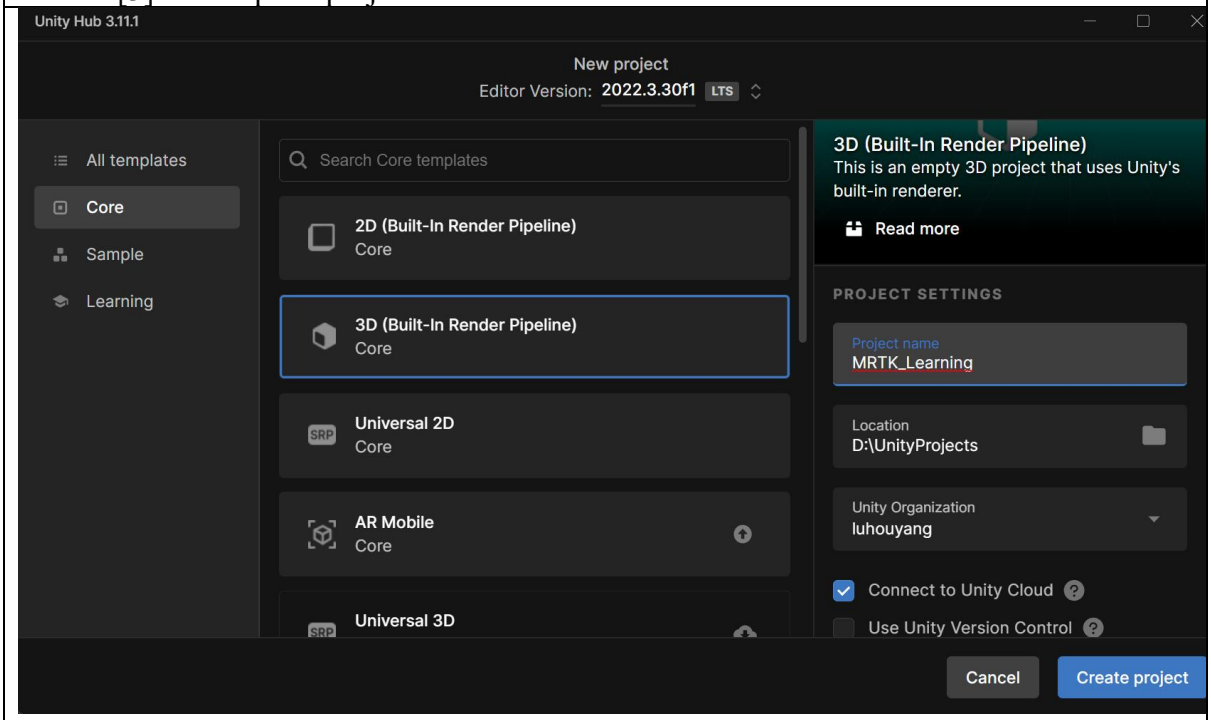




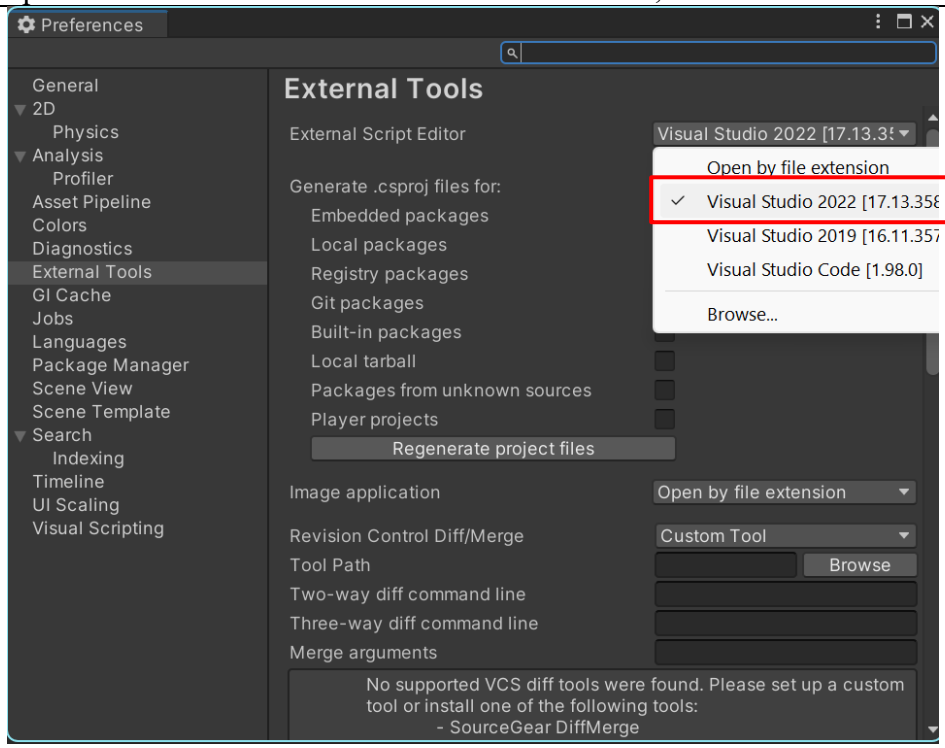
Download **Unity** [4]



Follow [3] to set up the project



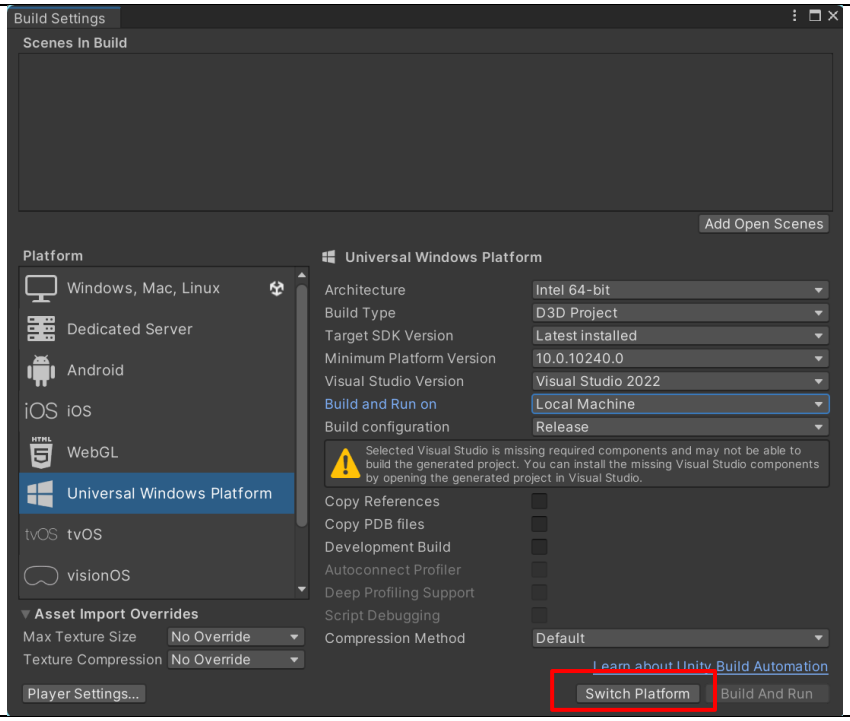
Check script editor at **Edit > Preferences > External Tools**, set to **Visual Studio 2022**



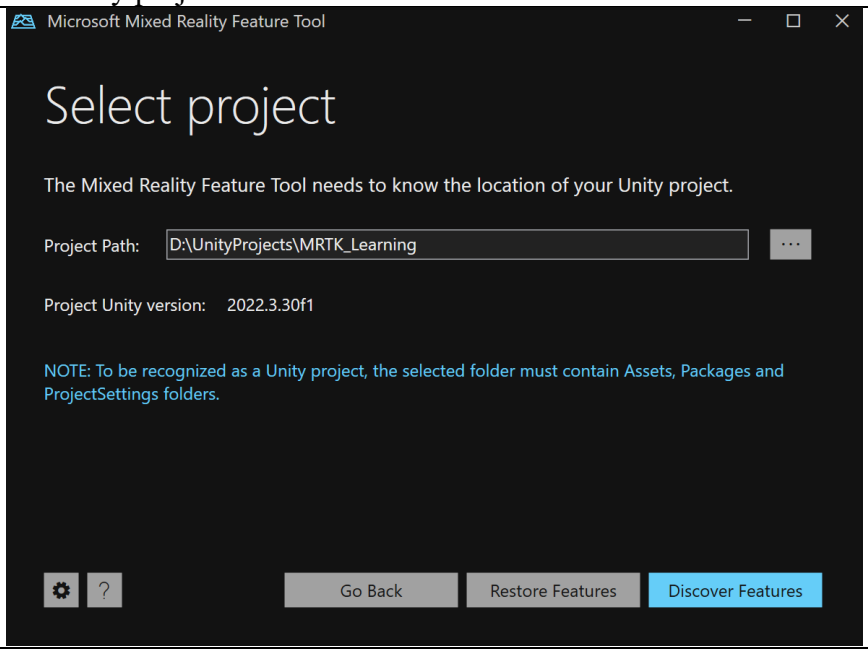
Select build target as **Universal Windows Platform** under **File > Build Settings > Platform**. Ensure the following:

- **Architecture**: ARM 64-bit
- **Build type**: D3D Project
- **Target SDK Version**: Latest Installed
- **Minimum Platform Version**: 10.0.10240.0
- **Visual Studio Version**: Visual Studio 2022

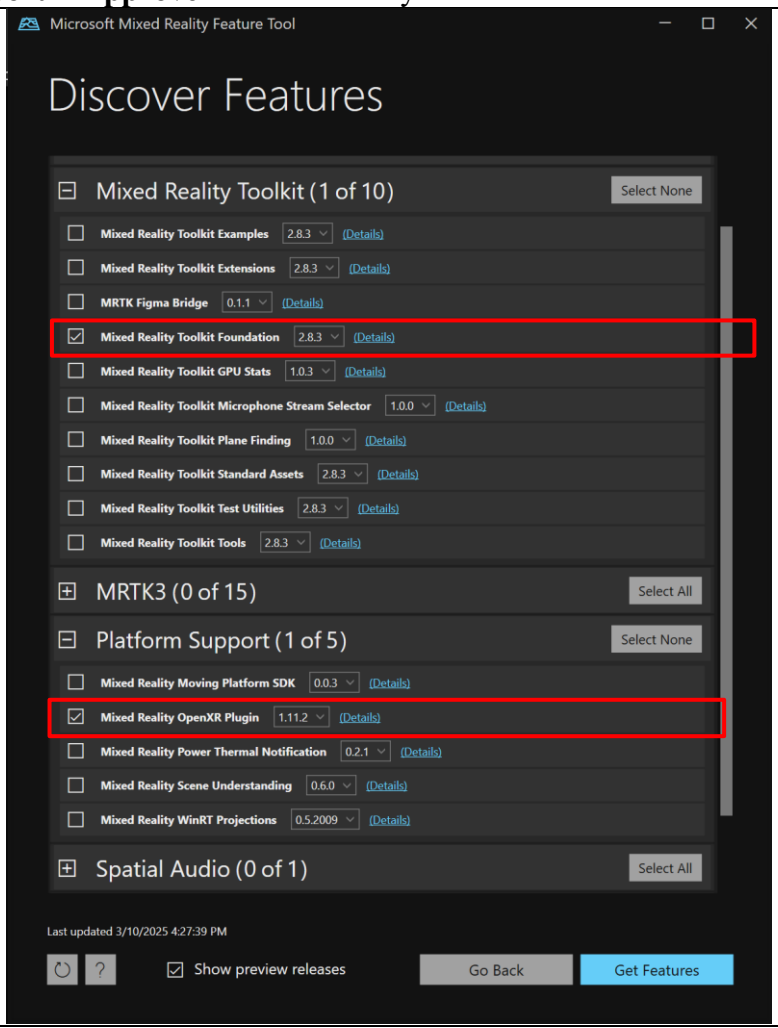
Click '**Switch Platform**'



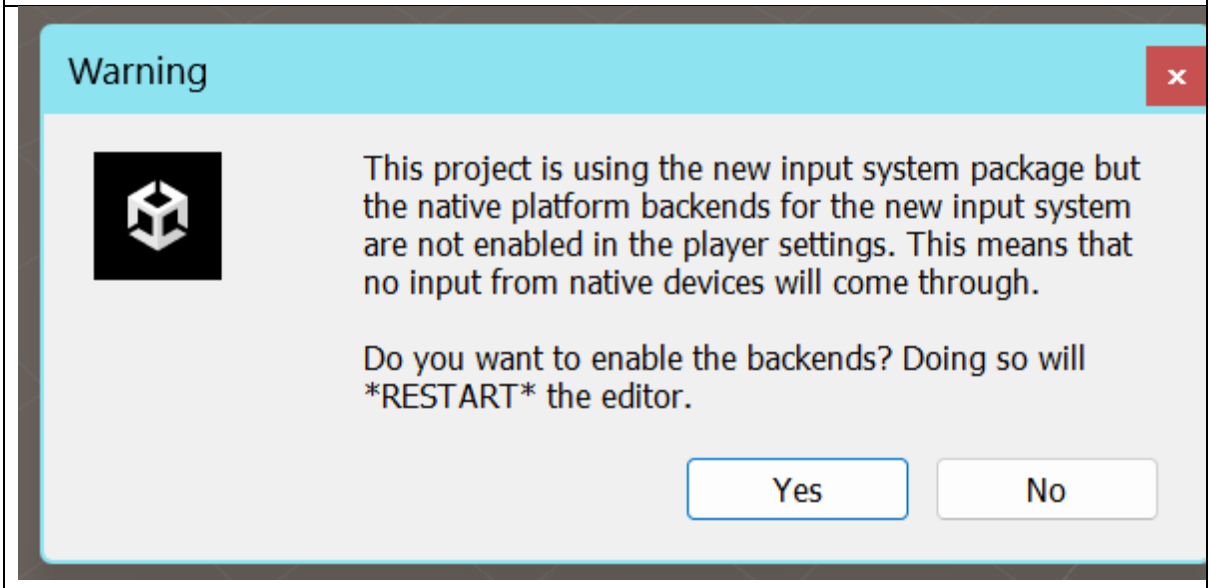
Download & run **MixedRealityFeatureTool.exe** save it to a new folder '**MRFeatureTool**' [8]. Select your Unity project & click '**Discover Features**'



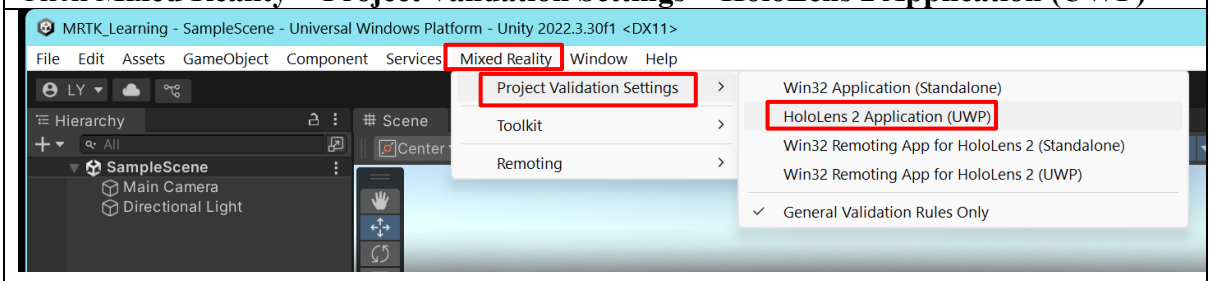
Select '**Mixed Reality Toolkit Foundation**' from '**Mixed Reality Toolkit (0 of 10)**' & **Mixed Reality OpenXR Plugin**' from '**Platform Support (0 of 5)**'. Click **Get Features** > **Validate** > **Import** > **Approve**. Return to Unity.



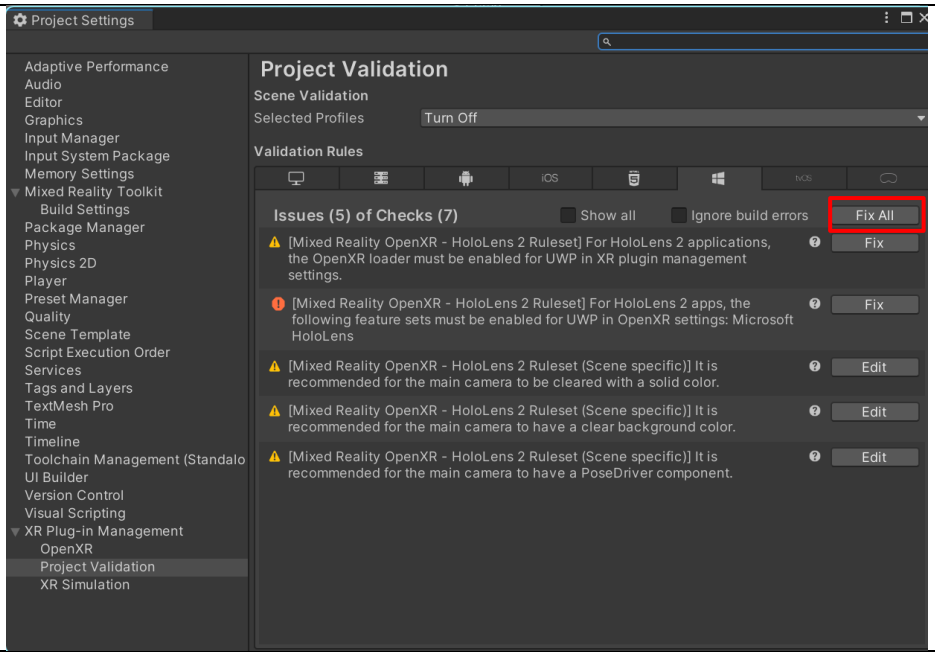
Click '**Yes**' to restart the editor.

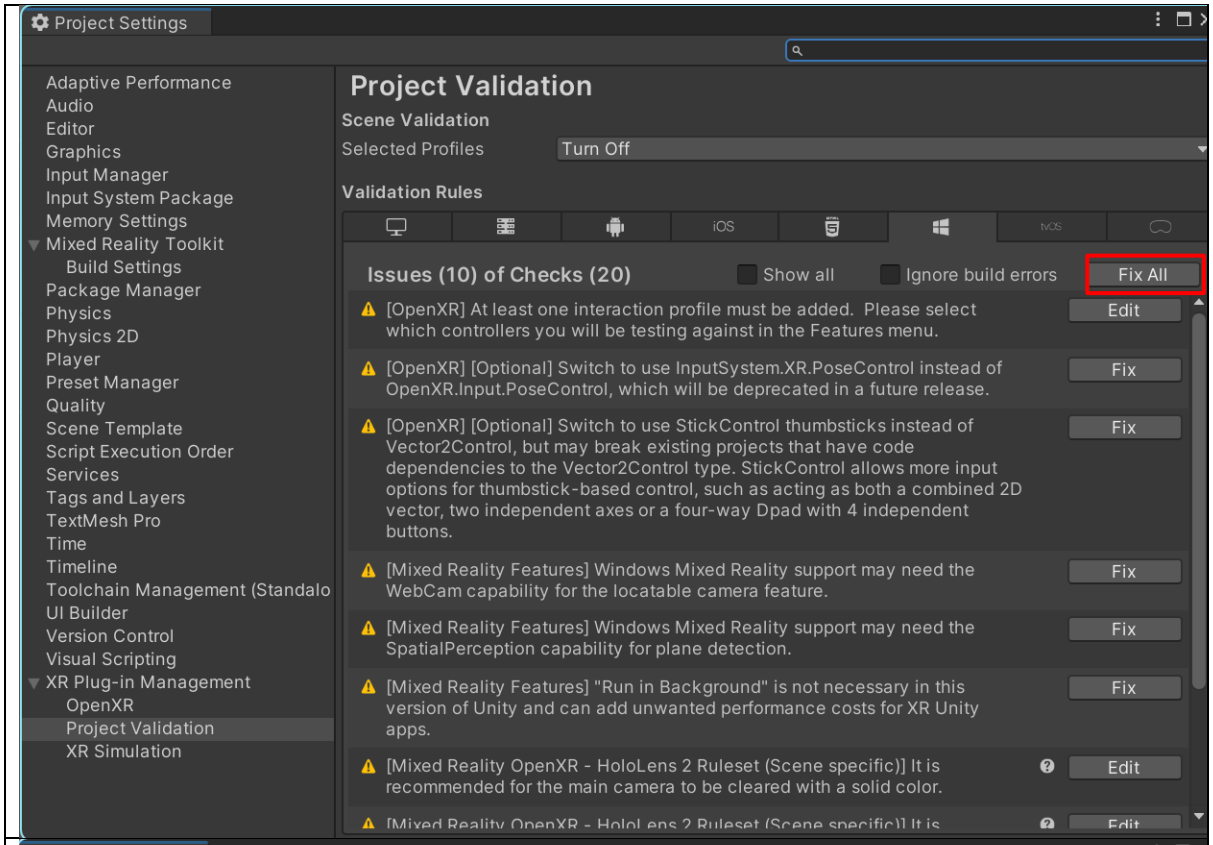


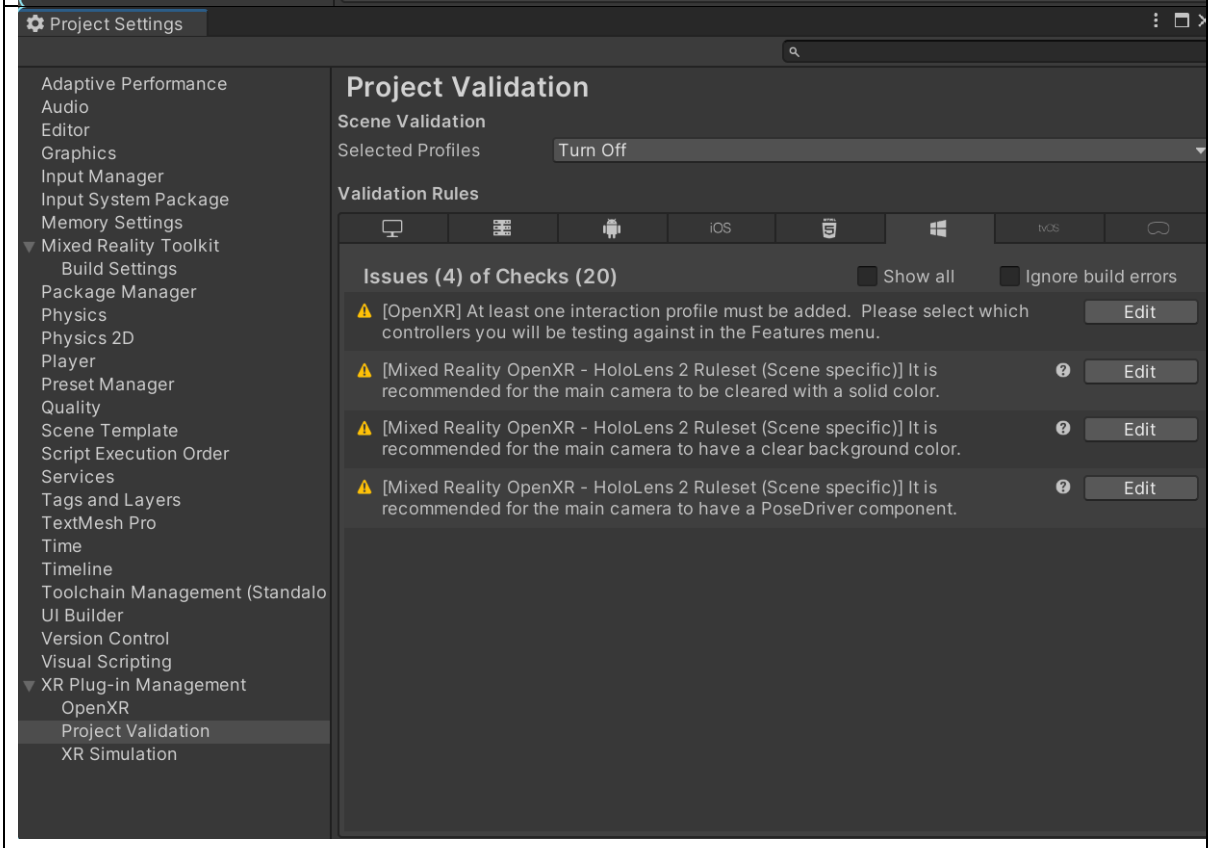
Click **Mixed Reality** > **Project Validation Settings** > **HoloLens 2 Application (UWP)**



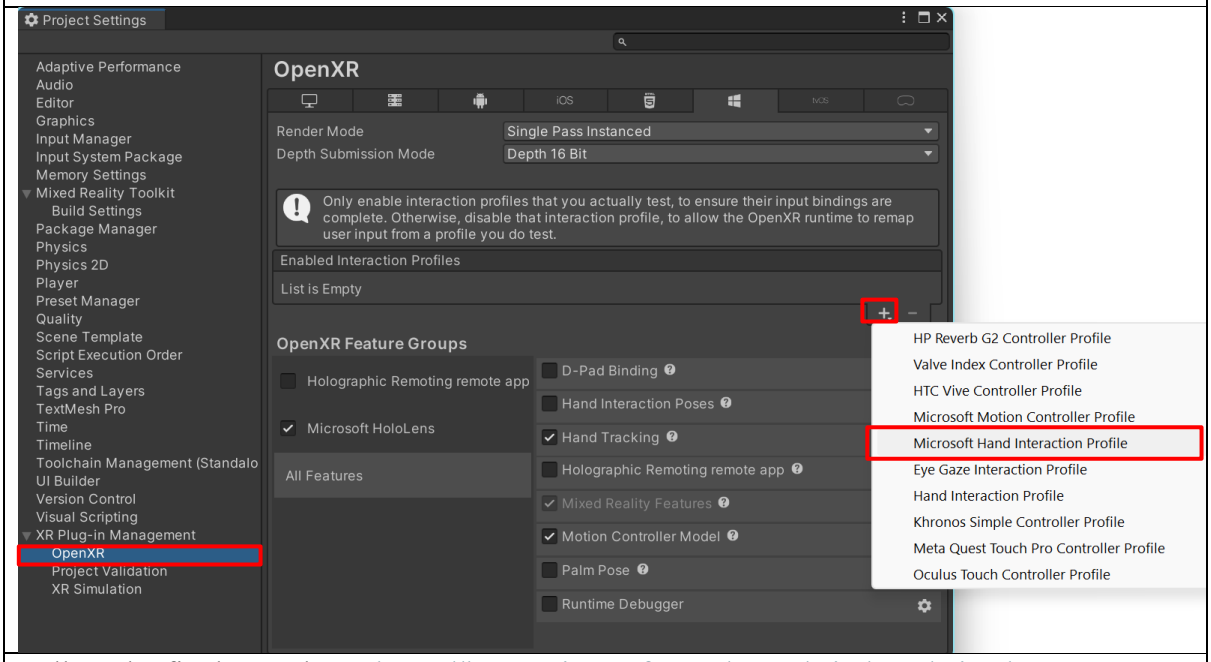
In the project validation window that pops up, make sure you're on the UWP platform tab (indicated by the Windows logo), and click Fix all to resolve the validation issues. Note that there may be issues that remain after clicking on Fix all. In that case, try clicking on Fix all again, ignore any issues that are marked "scene specific" (if any), and go through the rest of the issues (if any) to see if there are any suggested changes that you want to make.




Project Settings
Adaptive Performance
Audio
Editor
Graphics
Input Manager
Input System Package
Memory Settings
Mixed Reality Toolkit
Build Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Standalo
UI Builder
Version Control
Visual Scripting
XR Plug-in Management
OpenXR
Project Validation
XR Simulation
Project Validation
Scene Validation
Selected Profiles
Turn Off
Validation Rules
iOS
Issues (10) of Checks (20)
Show all
Ignore build errors
Fix All
[OpenXR] At least one interaction profile must be added. Please select which controllers you will be testing against in the Features menu.
Edit
[OpenXR] [Optional] Switch to use InputSystem.XR.PoseControl instead of OpenXR.Input.PoseControl, which will be deprecated in a future release.
Fix
[OpenXR] [Optional] Switch to use StickControl thumbsticks instead of Vector2Control, but may break existing projects that have code dependencies to the Vector2Control type. StickControl allows more input options for thumbstick-based control, such as acting as both a combined 2D vector, two independent axes or a four-way Dpad with 4 independent buttons.
Fix
[Mixed Reality Features] Windows Mixed Reality support may need the WebCam capability for the locatable camera feature.
Fix
[Mixed Reality Features] Windows Mixed Reality support may need the SpatialPerception capability for plane detection.
Fix
[Mixed Reality Features] "Run in Background" is not necessary in this version of Unity and can add unwanted performance costs for XR Unity apps.
Fix
[Mixed Reality OpenXR - HoloLens 2 Ruleset (Scene specific)] It is recommended for the main camera to be cleared with a solid color.
Edit


Project Settings
Adaptive Performance
Audio
Editor
Graphics
Input Manager
Input System Package
Memory Settings
Mixed Reality Toolkit
Build Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Tags and Layers
TextMesh Pro
Time
Timeline
Toolchain Management (Standalo
UI Builder
Version Control
Visual Scripting
XR Plug-in Management
OpenXR
Project Validation
XR Simulation
Project Validation
Scene Validation
Selected Profiles
Turn Off
Validation Rules
iOS
Issues (4) of Checks (20)
Show all
Ignore build errors
[OpenXR] At least one interaction profile must be added. Please select which controllers you will be testing against in the Features menu.
Edit
[Mixed Reality OpenXR - HoloLens 2 Ruleset (Scene specific)] It is recommended for the main camera to be cleared with a solid color.
Edit
[Mixed Reality OpenXR - HoloLens 2 Ruleset (Scene specific)] It is recommended for the main camera to have a clear background color.
Edit
[Mixed Reality OpenXR - HoloLens 2 Ruleset (Scene specific)] It is recommended for the main camera to have a PoseDriver component.
Edit

Open the **OpenXR** subsection. Select **Microsoft HoloLens**. Add the **Microsoft Hand Interaction Profile** as Interaction Profile



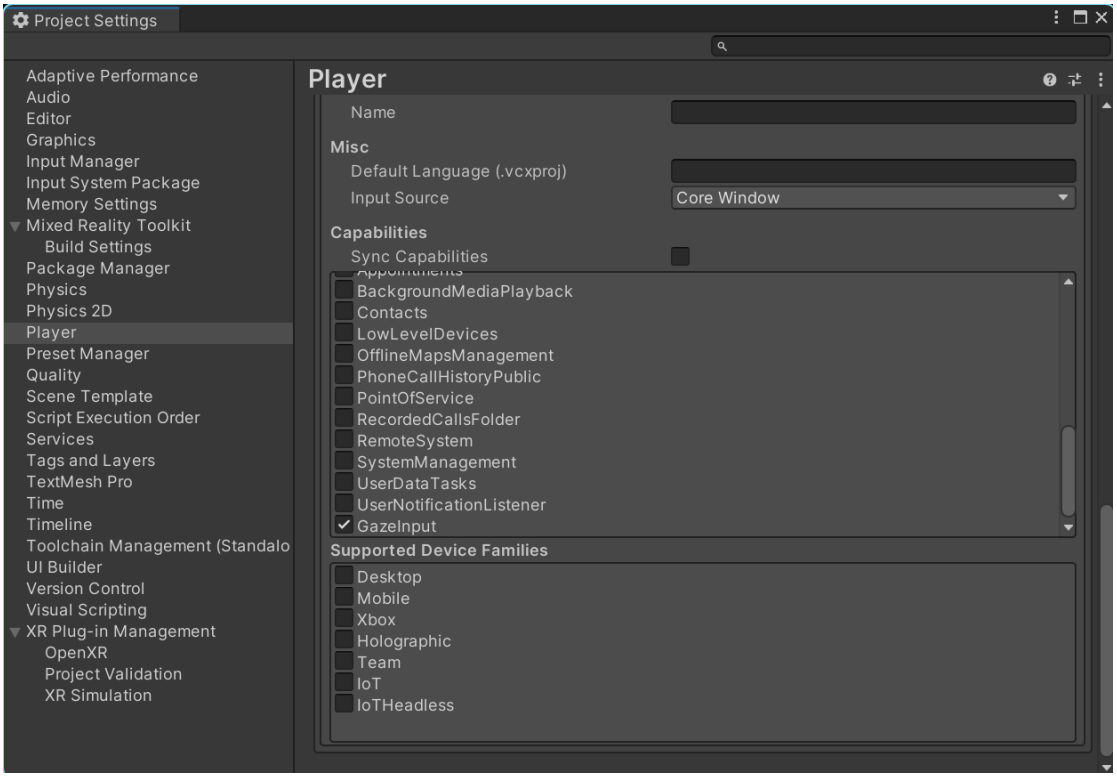
Follow the final steps here: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/new-openxr-project-with-mrtk#configure-player-settings

3 Integrating eye-tracking

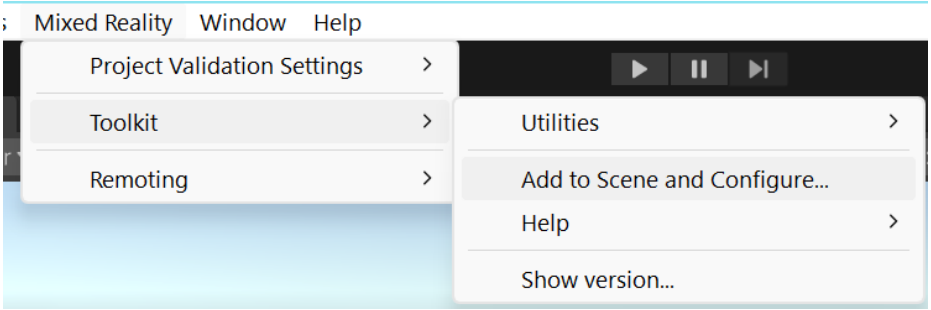
The fresh new scene setup outlined above with MRTK & OpenXR will be repeated for each following demo. The following section of the report covers the basics of eye-tracking integration with simulated objects. Following the implementation by Microsoft [9]: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-basic-setup?view=mrtkunity-2022-05

Eye-tracking requirements:

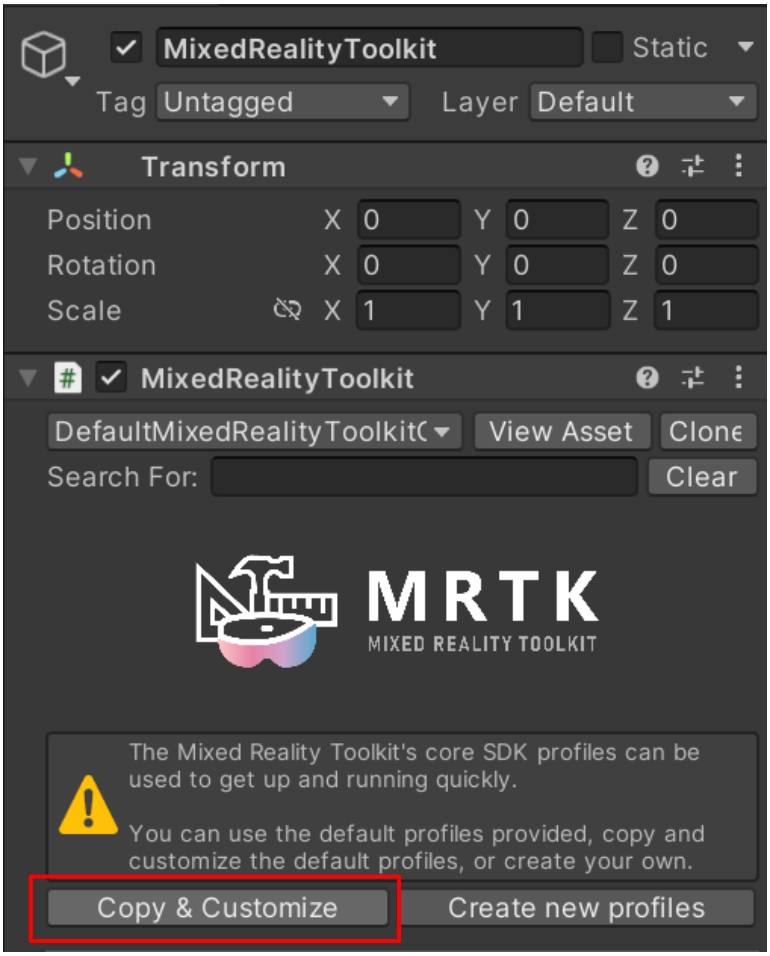
1. **GazeInput** must be enabled in application manifest.
    a. On Unity2022.3 LTS navigate to **Edit** > **Project Settings** > **Player** > **Publishing Settings** > **Capabilities** > (scroll down until you see **GazeInput**)

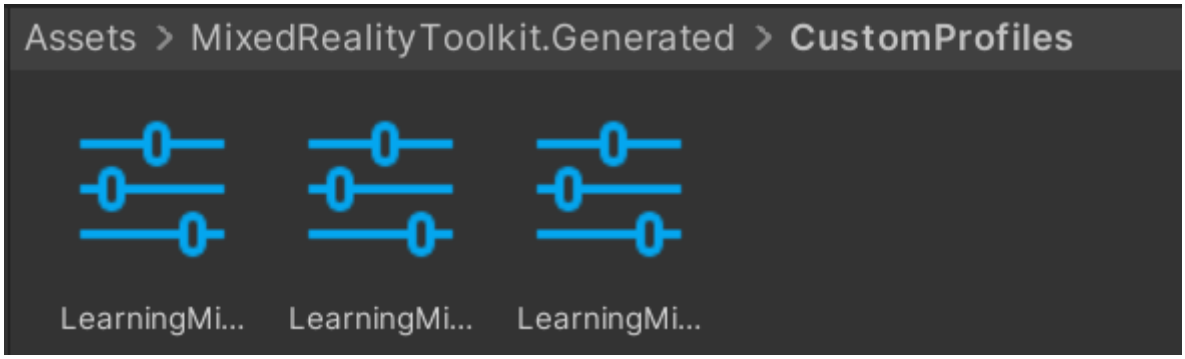


2. **Eye Gaze Data Provider** must be added to the input system.
    a. Click on **Mixed Reality** > **Toolkit** > **Add to Scene and Configure…**



b. Click on **Copy & Customize**, follow along here [9]: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-basic-setup?view=mrtkunity-2022-05#create-an-eye-gaze-data-provider





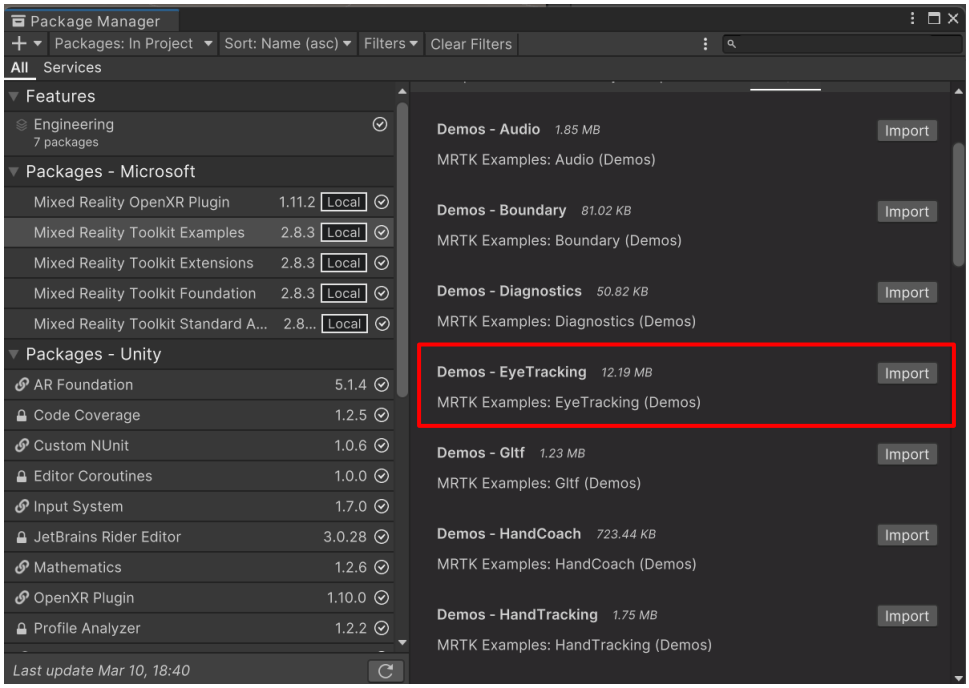
There should be at least 3 cloned scripts after the set up.

3. **HoloLens2** must be eye calibrated.
   a. *Can only be done after building to HoloLens2*

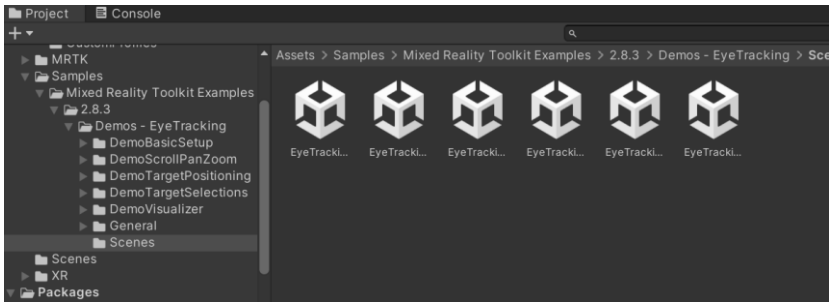
Simulations

You can simulate eye tracking input in the Unity Editor to ensure that events are correctly triggered before deploying the app to your HoloLens 2. The eye gaze signal is simulated by using the camera's location as eye gaze origin and the camera's forward vector as eye gaze direction [9]: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-basic-setup?view=mrtkunity-2022-05#simulating-eye-tracking-in-the-unity-editor

Short note on accessing eye tracking data [10]. Example scenes [11], follow the steps to add **Mixed Reality Toolkit Examples**: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/running-example-scenes?view=mrtkunity-2022-05



Scenes will be found in assets:



Demo overview [12]: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/example-scenes/eye-tracking-examples-overview?view=mrtkunity-2022-05#setting-up-the-mrtk-eye-tracking-samples

4 GitHub reference project

Eye tracking demo [13]: https://github.com/microsoft/MixedRealityToolkit-Unity/tree/main/Assets/MRTK/Examples/Demos/EyeTracking

5 Next steps

1. Learn the data attributes and pre- post-processing of eye-tracking data.
2. Draft of Unity application for HoloLens2 eye-tracking data collection.
3. Look into extended eye tracking capabilities of HoloLens2 [14].

6 Eye-tracking datasets

Eye-Tracking Technology [5]

Measuring point of gaze, position of eyes and collecting eye features. Eye-tracking data can be categorized to the essential ocular activity indicators – **fixation**, **saccades**, and **scan path**.

Fixation [5]

Eye motions stabilizing the retina above a stationary object of interest with duration of **100 – 400 ms**.

- Fairly centered
- Eye travels at low velocity
- Ocular drifts
- Ocular microtremor
- Micro saccades

Saccades [5]

Quick movements of both eyes used to reposition the fovea (middle part of retina) into a new location in the visual environment. Duration from **10 – 100 ms**.

- Reflexive and voluntary
- 4 types of saccade classification [6]
    - Predictive saccade
    - Anti saccade
    - Memory-guided saccade
    - Visually-guided saccade

Scan Path [5]

Direction taken by the eyes of the viewer when reading a text or observing a scene. Resulting from series of saccades and fixations.

Eye Tracker [5]

Device to detect eye movements and eye positions. Measure the visual attention of a subject by gathering data on eye movement when the subject observes a stimulus while operating on a task. There are 3 types of eye trackers.

- Eye-attached tracking

Using eye attachments (special contact lens). Measuring movement of eyes in horizontal, vertical, and torsional directions.

- Optical tracking

Determines position (retina/cornea) in real time by monitoring positions that are attached to the object. Location of reflex point is determined by means of a camera device, without direct contact with the eyes. Commonly used for gaze tracking (affordable, non-invasive).

- Measurement of electric potentials with electrodes

The eyes are the source of a permanent electrical field that can be observed in complete darkness even when eyes are closed. Example tracking method is electrooculogram (EOG) [7]. It is a technique used to measure the corneo-retinal standing potential that occurs between the forehead and back of the human eye. It is a very lightweight solution that needs only very low computing power. It also performs under various lighting conditions, and it can also be implemented as an integrated, self-contained wearable device (Sopic et al., 2018).

References

[1] sostel, “Eye tracking overview - Mixed Reality,” learn.microsoft.com. https://learn.microsoft.com/en-us/windows/mixed-reality/design/eye-tracking

[2] AMollis, “Choosing a Unity version and XR plugin - Mixed Reality,” Microsoft.com, Jan. 16, 2025. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/choosing-unity-version (accessed Mar. 10, 2025).

[3] Sean-Kerawala, “Set up a new OpenXR project with MRTK - Mixed Reality,” Microsoft.com, Nov. 02, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/new-openxr-project-with-mrtk (accessed Mar. 10, 2025).

[4] U. Technologies, “Get Started with Unity - Download the Unity Hub & Install the Editor,” unity.com. https://unity.com/download

[5] J. Z. Lim, J. Mountstephens, and J. Teo, “Eye-Tracking Feature Extraction for Biometric Machine Learning,” Frontiers in Neurorobotics, vol. 15, Feb. 2022, doi: https://doi.org/10.3389/fnbot.2021.796895

[6] N. N. J. Rommelse, S. Van der Stigchel, and J. A. Sergeant, “A review on eye movement studies in childhood and adolescent psychiatry,” Brain and Cognition, vol. 68, no. 3, pp. 391–414, Dec. 2008, doi: https://doi.org/10.1016/j.bandc.2008.08.025

[7] D. J. Creel, “The electrooculogram,” *Handbook of Clinical Neurology*, vol. 160, pp. 495–499, 2019, doi: https://doi.org/10.1016/B978-0-444-64032-1.00033-3

[8] “Download Mixed Reality Feature Tool from Official Microsoft Download Center,” Microsoft Store - Download Center, 2023. https://aka.ms/MRFeatureTool (accessed Mar. 10, 2025).

[9] lolambean, “Eye Tracking Basic Setup - MRTK 2,” Microsoft.com, Jan. 12, 2023. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-basic-setup?view=mrtkunity-2022-05 (accessed Mar. 10, 2025).

[10] lolambean, “Eye Tracking Eye Gaze Provider - MRTK 2,” Microsoft.com, Aug. 02, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-eye-gaze-provider?view=mrtkunity-2022-05 (accessed Mar. 10, 2025).

[11] lolambean, “Example scenes - MRTK 2,” Microsoft.com, Jun. 24, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/running-example-scenes?view=mrtkunity-2022-05 (accessed Mar. 10, 2025).

[12] lolambean, “Eye tracking examples overview - MRTK 2,” Microsoft.com, Jun. 24, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/example-scenes/eye-tracking-examples-overview?view=mrtkunity-2022-05#setting-up-the-mrtk-eye-tracking-samples (accessed Mar. 10, 2025).

[13] microsoft, “MixedRealityToolkit-Unity/Assets/MRTK/Examples/Demos/EyeTracking at main · microsoft/MixedRealityToolkit-Unity,” GitHub, 2016. https://github.com/microsoft/MixedRealityToolkit-Unity/tree/main/Assets/MRTK/Examples/Demos/EyeTracking (accessed Mar. 10, 2025).

[14] lolambean, “Extended eye tracking in Unity - Mixed Reality,” Microsoft.com, Nov. 29, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/extended-eye-tracking-unity (accessed Mar. 10, 2025).

[15] hamalawi, “Using the HoloLens Emulator - Mixed Reality,” learn.microsoft.com. https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-hololens-emulator