Developing Unity Applications for HoloLens, Magic Leap, and Quest 3

A Guide

Hello students and welcome to the Mixed Reality course!

We've written this guide for you to get a head start on developing your Mixed Reality application for HoloLens 2, Magic Leap 2, and Quest 3.

Please note that Microsoft, Magic Leap, and Unity **provide lots of documentation, guides, and examples online**. This guide is intended as a starting point for you, summarizing the main steps and providing references to useful online resources.

Throughout this guide, we have linked the official documentation on which we based each section. In case, there are any questions, **please have a look at the linked documentation** which may provide additional information. If this does not help either, reach out to us TAs to help you out.

Last but not least, software development is always a work in progress and APIs, versions, dependencies, and names change constantly. Thus, this guide also needs to be updated frequently. Please help us keep it up-to-date by letting us know about any errors, inconsistencies, outdated information, or other potential improvements that you may notice while following this guide.

Happy coding!
Your Mixed Reality TAs

Table of Content:

Preparation **4**
- Selection of a Development Environment 4
- Installing the Tools 4
- Additional Steps for Magic Leap 2 5

Setting up the Device **6**
- HoloLens 2 6
  - Windows Device Portal 6
  - Developer Mode 6
  - Emulator 6
- Magic Leap 2 6
  - Developer Mode 7
  - Simulator 7
- Quest 3 7
  - Meta Quest Link 7

Creating a Unity Project **7**
- Project Configuration for HoloLens 2 7
- Project Configuration for Magic Leap 2 10
- Project Configuration for Quest 3 13

Compilation and Deployment **17**
- Unity Simulation 17
- HoloLens 2 17
- Magic Leap 2 18
- Quest 3 19
- Device Simulators 20

Sharing Code with Unity **22**

Debugging **22**
- Debugging in Unity 22

Recording **22**
- HoloLens 2 22
- Magic Leap 2 23
- Quest 3 23

Additional Examples **24**
- On HoloLens 2 24
- On Magic Leap 2 24
- Quest 3 Passthrough 24

Server-Client Application Example **25**

Additional Links and Resources **26**
For Unity 26
For HoloLens 2 26
For Magic Leap 2 26
For Quest 3 26

Preparation

Selection of a Development Environment

The first step of implementing a Mixed Reality application for HoloLens 2, Magic Leap 2, and Quest 3 is the selection of a development environment or engine. Both devices offer **multiple options, such as Unity, Unreal Engine, or a native app**.

While we do not enforce any specific engine, we ***strongly recommend Unity*** due to the extensive community support and the availability of guides, documentation, and examples.

In fact, this guide only covers the basics of implementing a Unity application for HoloLens 2, Magic Leap 2, and Quest 3. Still, if you like to read up on your other options, check out Microsoft HoloLens 2, Magic Leap 2, and Meta Quest 3 documentation.

Installing the Tools

This section summarizes the installation instructions for Microsoft HoloLens 2, Magic Leap 2, and Meta Quest 3.

We recommend using **Unity 2022.3 LTS**, but more recent versions may be supported as well.

1. Download and install the latest Unity Hub. This guide uses Version 3. You'll have to create a free Unity account in case you don't have one yet.

2. When opening the Unity Hub for the first time, click on ***Skip Installation***. Then, click on ***Install Editor*** and select the latest version of Unity 2022.3 LTS.
   At the time of writing this guide, the latest version is 2022.3.43f1. On the next page, select the required modules in the depending on the device you are using:

   a. If you are developing for Magic Leap 2 or Quest 3, make sure to select the ***Android Build Support***, ***Android SDK & NDK Tools***, and ***OpenJDK*** modules.

   b. If you are developing for HoloLens 2, make sure to select the ***Universal Windows Platform Build Support*** and the ***Windows Build Support (IL2CPP)*** modules.

   c. If you don't have Visual Studio installed, feel free to select the tick box in the ***developer*** section. This will automatically start the installation process for Visual Studio 2022. When installing Visual Studio, make sure you install the required

components listed in the next step.

3. Install Visual Studio 2022 either via the Unity Hub or by downloading the installer here. The ***community version*** is sufficient.

    a. In the ***Workloads*** tab, make sure to install the following workloads:
        i. ***Game development with Unity***

    b. For **HoloLens 2** development, also install the following workloads:
        i. ***.NET desktop development***
        ii. ***Desktop development with C++***
        iii. ***Windows application development.*** In the installation details, also select the following components:
            1. ***Universal Windows Platform tools***
            2. ***C++ (v143) Universal Windows Platform tools***

Additional Steps for Magic Leap 2

This guide is based on the official installation guide.

1. Download and install the latest Magic Leap Hub. This guide uses Version 3.
2. Open the ***Magic Leap Hub***, and select the ***Packages*** tab on the left navigation bar.
3. Install the latest ***Unity Package*** and ***Unity MRTK3*** packages. This guide uses Unity Package ***version 2.4.0*** and Unity MRTK3 ***version 1.2.0***.

Setting up the Device

We will hand you the devices in a factory reset state, you may have to create your own accounts to use the devices.

HoloLens 2

Windows Device Portal

The official documentation can be found here.

HoloLens 2 has a web portal that allows you to record the experience, upload or download files, and provides several other features.

We access the ***Windows Device Portal*** through a URL link that you can find on the bottom of the HoloLens ***Settings*** > ***For developers*** page within the HoloLens. Enter the URL on your favorite web browser on your laptop.

If you forgot the password, you can simply enter a wrong password three times and then re-pair the device to set a new one.

Developer Mode

Go to HoloLens ***Settings*** > ***For developers***, enable ***Use developer features***, ***Device discovery***, and ***Device Portal***.

Emulator

For a faster iteration cycle, you can also develop using an ***Emulator***, so you don't have to flash your app to the HoloLens device every single time. Documentation can be found here.

Magic Leap 2

You can follow the steps in this link to turn on your device and get everything paired up.

Magic Leap Hub

Magic Leap Hub 3 groups useful developer tools and utilities in one convenient place.

Good to know: Resources that are part of the Magic Leap Hub are: Device management (including OS and packages), Apps, Video/image capture, Streaming, Rendering, Logs for troubleshooting, Settings, Commandline, and Documentation.

The official documentation can be found here.

Developer Mode

Please follow this guide to activate the developer mode on Magic Leap 2.

Simulator

For Testing your application without deploying it to Magic Leap 2, ***Application Simulator*** can be used. Please follow the steps to setup Application Simulator here.

Quest 3

Please follow this guide to activate the developer mode on the Meta Quest 3.

The official documentation can be found here.

The Meta Quest Link can be used to connect your device to your PC to aid you in your development process. The link enables you to run and render your app on a remote machine (e.g. your laptop) and stream the rendered images to the Quest 3. This may be useful for applications with high-quality graphics.

Creating a Unity Project

Project Configuration for HoloLens 2

The following steps are based on the instructions in the following links: link1, link2.

1. In the Unity Hub, create a new project using the provided *Universal 3D* template.

2. Download the Mixed Reality Feature Tool.

3. Start the Mixed Reality Feature Tool and select your Unity project directory, similar instructions can be found here. Select the packages listed below. This guide uses ***version 4.0.0-pre.1*** of all ***MRTK3 packages*** and ***version 1.11.1*** of the ***Mixed Reality OpenXR Plugin***.

    a. From the *MRTK3* section, select the following checkboxes:

        i. ***MRTK Core Definitions***
        ii. ***MRTK Input***
        iii. ***MRTK Spatial Manipulation***
        iv. ***MRTK UX Components***
        v. ***MRTK UX Components (Non-Canvas)***
        vi. ***MRKT UX Core Scripts***

    b. From the *Platform Support* section, select the following checkboxes:

        i. ***Mixed Reality OpenXR Plugin***

    c. Click on ***Get Features***, ***Validate***, ***Import***, and ***Approve*** to add ***MRTK3*** to your project.

        Unity should detect the changes automatically and install the new packages.

    d. When asked to restart the editor, click ***Yes***.

    e. For more information on which features each package provides, see this page.

4. In Unity, open the ***Package Manager*** (Window > Package Manager) and install ***Unity glTFast***. Click on the **+** icon in the top-left corner, then select ***Add package by name…*** and enter ***com.unity.cloud.gltfast*** in the name field. This guide uses ***version 6.7.1***.

5. Under ***Edit > Project Settings > MRTK3***, select the ***Universal Windows Platform*** tab and make the following changes:

    a. Set the Profile to ***MRTKProfile*** by clicking on the circle icon on the right. In the ***Select MRTK Profile*** dialog window, you may have to toggle the package visibility using the button in the top right corner to show all options. Alternatively, the profile can be found in the ***Project*** window under `Packages/MRTK Core Definitions/Configuration/Default Profiles/MRTKProfile.asset`

    b. Enable the ***MRTK Hands Aggregator Subsystem, Subsystem for OpenXR Hands API,*** *and* ***Subsystem for Hand Synthesis.***

6. Under ***Edit > Project Settings > XR Plug-in Management***, select the **Universal Windows Platform** tab, tick **OpenXR** and the ***Microsoft HoloLens feature group***. Remove all other ticks.

7. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, make sure that the ***Enabled Interaction Profiles*** contains exactly the following items
    i. ***Eye Gaze Interaction Profile***
    ii. ***Microsoft Hand Interaction Profile***
    iii. ***Microsoft Motion Controller Profile***

8. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, set the ***Depth Submission Mode*** to ***Depth 16-bit***.

9. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, in the ***OpenXR Feature Groups***, ensure the following features are enabled:
    i. ***Hand Tracking***
    ii. ***Mixed Reality Features***
    iii. ***Motion Controller Model***

    a. You may select additional features that you need in your project. Google them if you need specific ones :)

10. *Under* ***Edit > Project Settings > XR Plug-in Management > Project Validation***, click the ***Fix All*** button to solve some configuration issues. Some issues may persist, but you can ignore them for now.

11. Under ***Edit > Project Settings > Player***, set a nice name for your new application.
    a. Choose a ***Product Name***. This will be the name of our application.
    b. Set the ***Company Name*** to your team name or *MixedRealityETHZ*.

12. Let's create a new scene:

    a. ***Right-click*** on the ***SampleScene*** in the ***Hierarchy*** **window** and select ***Add New Scene***.

    b. Delete the ***Main Camera***, as we will replace it with the ***MRTK XR Rig***. You may also remove the ***SampleScene*** by ***right-clicking*** it and selecting ***Remove Scene***.

    c. Locate the ***MRTK XR Rig*** under `Packages/MRTK Input/Assets/Prefabs/` and add it to the scene by dragging the item from the *Project* window into the ***Hierarchy*** window.

    d. ***Right-click*** in the Hierarchy window, and select ***3D Object > Cube***.

    e. To ensure the cube is visible in front of the user upon start, set its Position to Y=1.6m and Z=0.5m, and set the scale to 0.2m in all three dimensions.

    f. To let the user interact with the cube, attach an ***Object Manipulator*** component to it. To do so, select the ***Cube*** in the ***Hierarchy window***, then click on ***Add Component*** in the ***Inspector window*** and search for ***Object Manipulator***.

    g. Save the scene

13. To deploy the app, follow the instructions in the **Compilation and Deployment** Section below.

Project Configuration for Magic Leap 2

The following steps are taken from here.

1. In the Unity Hub, create a ***new project*** using the provided ***Universal 3D template***.

2. Download the Mixed Reality Feature Tool.

3. Start the Mixed Reality Feature Tool and select your Unity project directory. Select the packages listed below. This guide uses ***version 4.0.0-pre.1*** of all packages.

    a. From the ***MRTK3*** section, select the following checkboxes:

i. ***MRTK Core Definitions***
ii. ***MRTK Input***
iii. ***MRTK Spatial Manipulation***
iv. ***MRTK UX Components***
v. ***MRTK UX Components (Non-Canvas)***
vi. ***MRKT UX Core Scripts***

b. Click on ***Get Features***, ***Validate***, ***Import***, and ***Approve*** to add ***MRTK3*** to your project. Unity should detect the changes automatically and install the new packages.

c. When asked to restart the editor, click ***Yes***.

d. For more information on which features each package provides, see this page.

4. Configure your new project for Magic Leap using the ***Magic Leap Setup Tool***:

   a. Download the Magic Leap Setup Tool. Click on ***Add to My Assets***, then on ***Open in Unity***. This guide uses ***version 2.0.12***.

   b. The asset should open in the **Package Manager** inside the **Unity Editor**. Click ***Download***, ***Import***, ***Select All***, ***Import*** to add it to the project. When asked, choose ***OpenXR***.

   c. ***Magic Leap Project Setup Tool*** should pop up in a new window. Locate ***Magic Leap SDK*** folder in `C:\Users\<YOURUSER>\MagicLeap\mlsdk\<VERSION>`
   This guide uses ***Magic Leap SDK version 1.9.0***.

   d. Then click on ***Fix Setting*** and set the *build target* to *Android*.

   e. Click on ***Apply All*** to complete the remaining steps. When asked, use the ***Magic Leap registry***. This step may take a while; wait until all scripts have been compiled. When asked to restart the editor, click ***Yes***. At the end of this step, all buttons should turn green. If not, try clicking ***Apply All*** again.

5. Inside the ***Package Manager*** (***Window > Package Manager***), select the + icon in the top-left corner, then select ***Add package from tarball***

   a. Locate and import the `com.magicleap.mrtk3-[VERSION].tgz` package from `MagicLeap/tools/unity/mrtk3/`. This guide uses ***version 1.2.0***.

6. Under ***Edit > Project Settings > XR Plug-in Management***, select the ***Android tab***. In the Plug-in Providers, tick the ***OpenXR*** and ***Magic Leap 2 feature group*** items and

remove all other ticks.

7. Under ***Project Settings > XR Plug-in Management > OpenXR***, make sure that the ***Enabled Interaction Profiles*** contain exactly the following items:

    i. ***Magic Leap 2 Controller Interaction Profile***
    ii. ***Hand Interaction Profile***
    iii. ***Eye Gaze Interaction Profile***

    a. You may select additional features that you need in your project. Google them if you need specific ones :)

8. Under ***Project Settings > XR Plug-in Management > OpenXR***, in the ***OpenXR Feature Groups***, tick the ***Magic Leap 2 box*** in the left column and the ***Hand Tracking Subsystem*** in the right column. You may select additional Magic Leap 2 features that you need in your project.

9. Under ***Project Settings > MRTK3***, set the Profile to ***MRTKProfile-MagicLeap-OpenXR*** by clicking on the circle icon on the right. In the ***Select MRTK Profile*** dialog window, you may have to toggle the package visibility using the button in the top right corner to show all options. Alternatively, the profile can be found in the *Project* window under

    ```
    Packages/Magic Leap MRTK3
    /Runtime/XRProviders/OpenXR/Configuration/Default
    Profiles/MRTKProfile-MagicLeap-OpenXR.asset
    ```

10. Under ***Project Settings > MagicLeap > Permissions***, select the permissions that your app will require. For the example in this guide, select the following permissions:

    i. **EYE_TRACKING**
    ii. **VOICE_INPUT**
    iii. **SPATIAL_MAPPING**
    iv. **HAND_TRACKING**

    To figure out which permissions are required, please check out this page and this list of permissions needed for each API function.

11. Under ***Project Settings > Player > Android tab > Other Settings > Configuration***, look for the ***Active Input Handling*** property and change its value to ***Input System Package (new)***. Unity will restart to apply this change.

12. Under ***Edit > Project Settings > Player***, set a nice name for your new application.

    a. Choose a ***Product Name***. This will be the name of our application.

b. Set the ***Company Name*** to your team name or *MixedRealityETHZ*.

13. Let's create a new scene:
    a. ***Right-click*** on the ***SampleScene*** in the ***Hierarchy window*** and select ***Add New Scene***.

    b. Delete the ***Main Camera***, as we will replace it with the ***MRTK XR Rig***. You may also remove the ***SampleScene*** by ***right-clicking*** it and selecting ***Remove Scene***.

    c. Locate the ***MRTK XR Rig - MagicLeap - OpenXR*** under `Packages/Magic Leap MRTK3/Runtime/XRProviders/OpenXR/Prefabs/MRTK_Variants/` and add it to the scene by dragging the item from the ***Project window*** into the ***Hierarchy window.***

    d. ***Right-click*** in the ***Hierarchy window***, and select ***3D Object > Cube***.

    e. To ensure the cube is visible in front of the user upon start, set its Position to Y=1.6m and Z=0.5m, and set the scale to 0.2m in all three dimensions.

    f. To let the user interact with the cube, attach an ***Object Manipulator*** component to it. To do so, select the ***Cube*** in the ***Hierarchy window***, then click on ***Add Component*** in the ***Inspector window*** and search for ***Object Manipulator***.

    g. Save the scene

14. To deploy the app, follow the instructions in the **Compilation and Deployment** Section below.

Project Configuration for Quest 3

1. In the ***Unity Hub***, create a new project using the provided ***Universal 3D*** **template**.

2. Download the Mixed Reality Feature Tool.

3. Start the Mixed Reality Feature Tool and select your Unity project directory. Select the packages listed below. This guide uses ***version 4.0.0-pre.1*** of all packages.

    a. From the *MRTK3* section:
        i. MRTK Core Definitions
        ii. MRTK Input

iii. ***MRTK Spatial Manipulation***
iv. ***MRTK UX Components***
v. ***MRTK UX Components (Non-Canvas)***
vi. ***MRKT UX Core Scripts***

b. Click on ***Get Features***, ***Validate***, ***Import***, and ***Approve*** to add **MRTK3** to your project.

   Unity should detect the changes automatically and install the new packages.

c. When asked to restart the editor, click ***Yes***.

d. For more information on which features each package provides, see this page.

4. In Unity, open the **Package Manager** (Window > Package Manager) and install the packages listed below. You may need to select ***Packages: Unity Registry*** in the top-left corner to find them.

   a. Search for the ***OpenXR Plugin***. This guide uses **version 1.12.0**. During the installation, the ***Project Settings*** window may appear. You can ignore the listed project validation issues for now.

   b. Search for ***Unity OpenXR Meta***. This guide uses **version 1.0.1**.

   c. Install ***Unity glTFast***. Click on the **+ icon** in the top-left corner, then select ***Add package by name…*** and enter ***com.unity.cloud.gltfast*** in the name field. This guide uses **version 6.7.1**.

5. Under ***Edit > Project Settings > MRTK3***, select the ***Android*** tab and make the following changes:

   a. Set the Profile to ***MRTKProfile*** by clicking on the circle icon on the right. In the *Select MRTK Profile* dialog window, you may have to toggle the package visibility using the button in the top right corner to show all options. Alternatively, the profile can be found in the *Project* window under `Packages/MRTK Core Definitions/Configuration/Default Profiles/MRTKProfile.asset`

   b. Enable the ***MRTK Hands Aggregator Subsystem***

   c. Enable the ***Subsystem for Unity XR Hands API***

   d. Enable the ***Subsystem for Hand Synthesis***

6. Under ***Edit > Project Settings > XR Plug-in Management***, select the ***Android*** tab, tick ***OpenXR*** and the ***Meta Quest feature group***. Remove all other ticks.

7. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, make sure that the ***Enabled Interaction Profiles*** contains exactly the following items

    a. ***Oculus Touch Controller Profile***
    b. ***Hand Interaction Profile***

8. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, in the ***OpenXR Feature Groups***, tick the following features:

        i. ***Hand Tracking Subsystem***
        ii. ***Meta Quest Support***
        iii. ***Meta Quest: AR Camera (Passthrough)***

    b. You may select additional features that you need in your project. Google them if you need specific ones :)

9. Under ***Edit > Project Settings > XR Plug-in Management > Project Validation***, click the ***Fix All*** button to solve some configuration issues. Some issues may persist, but you can ignore them for now.

10. Under ***Edit > Project Settings > Player > Android tab > Other Settings > Configuration***, make the following adjustments:

    a. Change the ***Scripting Backend*** to ***IL2CPP***.
    b. Change the ***Target Architectures*** to ***ARM64***.
    c. Change the ***Active Input Handling*** to ***Input System Package (new)***. Unity will restart to apply this change.

11. Under ***Edit > Project Settings > Player***, set a nice name for your new application.

    a. Choose a ***Product Name***. This will be the name of our application.
    b. Set the ***Company Name*** to your team name or *MixedRealityETHZ*.

12. Let's create a new scene:

    a. Right-click on the **SampleScene** in the ***Hierarchy*** **window** and select ***Add New Scene***.

    b. Delete the **Main Camera**, as we will replace it with the **MRTK XR Rig**. You may also remove the **SampleScene** by right-clicking it and selecting ***Remove Scene***.

c. Locate the ***MRTK XR Rig*** under `Packages/MRTK Input/Assets/Prefabs/` and add it to the scene by **dragging the item** from the ***Project* window** into the ***Hierarchy* window**.

d. Right-click in the **Hierarchy window**, and select ***3D Object* > *Cube***.

e. To ensure the cube is visible in front of the user upon start, set its **Position** to Y=1.6m and Z=0.5m, and set the **scale** to 0.2m in all three dimensions.

f. To let the user interact with the cube, attach an ***Object Manipulator*** component to it. To do so, select the Cube in the ***Hierarchy* window**, then click on ***Add Component*** in the ***Inspector* window** and search for ***Object Manipulator***.

g. Save the scene

13. To deploy the app, follow the instructions in the **Compilation and Deployment** Section below.

Compilation and Deployment

This section details how to run your app on the device. All devices should support both USB cable and WiFi deployment, but in our experience, cable deployment is in general faster. If you would like to deploy your app over WiFi, or even just connect to your device from your laptop over WiFI, make sure your network supports this connection (i.e. firewalls, specific IP addresses, etc).

Unity Simulation

Just click on the Play button within Unity. You can also simulate your hand interactions in Unity, more here.

HoloLens 2

Unity applications can easily be deployed to the HoloLens 2 via USB or Wifi. However, the build and deploy process takes several minutes, which becomes time consuming with rapid prototyping. In this earlier development phase, it's recommended to use Holographic Remoting instead!

1. Open the Build Settings via ***File > Build Settings***

    a. Make sure that your scene(s) are listed under ***Scenes In Build***. Remove any unwanted scenes.

    b. Ensure that the ***Universal Windows Platform*** is selected.

    c. Set the ***Architecture*** to ***ARM 64-bit***.

    d. ***Build*** the project. At the first run Unity will ask you for a build directory. Feel free to create a Build folder in the project's main directory, or wherever you want to put the builds.

2. After the project was built, navigate to the built directory and open the **.sln solution file** in Visual Studio.

    a. Switch to ***Release*** mode.

    b. Switch to the ARM64 platform.

3. Deploy the app

a. If you want to deploy **via Wifi**, select ***Remote Machine*** as the target. Under ***Project > Properties > Configuration Properties > Debugging***, set the ***Machine Name*** to your device's Wifi IP.

b. If you want to deploy **via USB cable**, select ***Device*** as the target.

c. Deploy via ***Build > Deploy Solution***.

4. Once these settings have been adjusted, you can deploy your app via the ***Build And Run*** button or via ***File > Build And Run***.

5. After successful deployment, you should see the following experience:

    a. On the device, you should first see the "Made with Unity" splash screen, followed by a white cube floating in front of you, if you're still using the "cube app" we made previously.

    b. Look at your hands. You should see two rays shooting out of your index fingers into the distance. If you touch your thumb and index finger (this is the **pinch gesture**), you should see white outlines around both fingers.

    c. Look at the controller. You should see a virtual controller model superimposed onto the real one. Also, a ray should shoot out into the distance.

    d. Point the controller ray or your hand rays at the white cube in front of you. A white circle appears at the intersection of the ray with the cube. On the controller, you can press the trigger with your index finger to grab the cube. When using the hands, do the pinch gesture to grab it. While grabbing the cube, you can move and rotate it.

    e. To scale the cube, you can grab the cube using both hands at the same time. Then, pull your hands apart to scale it up, or move them closer together to scale it down.

Magic Leap 2

Unity applications can easily be deployed to the Magic Leap via USB or Wifi. In both cases, the device should automatically be detected in the **Magic Leap Hub** and in **Unity**.

1. Open the ***Build Settings*** via ***File > Build Settings***…

a. Make sure that your scene(s) are listed under ***Scenes In Build***. Remove any unwanted scenes.

b. Ensure that the ***Android*** platform is selected.

c. Ensure that your ***Magic Leap 2*** device is listed in the ***Run Device*** drop-down menu.

2. Once these settings have been adjusted, you can deploy your app via the ***Build And Run*** button or via ***File > Build And Run***.

3. After successful deployment, you should the the following experience:

    a. On the device, you should first see the “Made with Unity” splash screen, followed by a white cube floating in front of you, if you’re still using the “cube app” we made previously.

    b. Look at your hands. You should see two rays shooting out of your index fingers into the distance. If you touch your thumb and index finger (this is the **pinch gesture**), you should see white outlines around both fingers.

    c. Look at the controller. You should see a virtual controller model superimposed onto the real one. Also, a ray should shoot out into the distance.

    d. Point the controller ray or your hand rays at the white cube in front of you. A white circle appears at the intersection of the ray with the cube. On the controller, you can press the trigger with your index finger to grab the cube. When using the hands, do the pinch gesture to grab it. While grabbing the cube, you can move and rotate it.

    e. To scale the cube, you can grab the cube using both hands at the same time. Then, pull your hands apart to scale it up, or move them closer together to scale it down.

If the Magic Leap 2 is not automatically detected, look up its address under ***Settings > About > IP address***. Open the ***Magic Leap hub***, click on ***Connect via Wi-Fi*** and enter the IP address. Then, in Unity under ***File > Build Settings***, click the ***Refresh*** button next to the **Run Device** property, or manually enter the IP again. We tested this on the eth-5 Wifi.

Quest 3

Connect the Quest 3 to your computer via USB.

1. Open the **Build Settings** via ***File > Build Settings***…

    a. Make sure that your scene(s) are listed under ***Scenes In Build***. Remove any unwanted scenes.

    b. Ensure that the ***Android*** platform is selected.

    c. Ensure that your ***Oculus Quest 3*** device is listed in the ***Run Device*** drop-down menu.

2. Once these settings have been adjusted, you can deploy your app via the ***Build And Run*** button or via ***File > Build And Run***.

3. After successful deployment, you should have the following experience:

    a. On the device, you should first see the “Made with Unity” splash screen, followed by a white cube floating in front of you, if you’re still using the “cube app” we made previously.

    b. Look at your hands. After a moment you should see virtual hands aligned with your real hands.

    c. Pick up the controllers. You should see virtual controller models superimposed onto the real ones.

    d. Point the controller ray or your hand rays at the white cube in front of you. A white circle appears at the intersection of the ray with the cube. On the controller, you can press the trigger with your index finger to grab the cube. When using the hands, do the pinch gesture to grab it. While grabbing the cube, you can move and rotate it.

    e. To scale the cube, you can grab the cube using both hands at the same time. Then, pull your hands apart to scale it up, or move them closer together to scale it down.

Sharing Code with Unity

Since you’ll be working in teams, it’s important to be able to work on the same project base easily. This is where ***Unity Version Control*** comes in handy.

Debugging

Debugging in Unity

The workflow for debugging in Unity can be different depending on what your needs are. Here are some options:

- Debugging directly in ***Unity Play mode*** by playing the app
- Debugging with ***Holographic Remoting***
- Simulation of Hololens: ***HoloLens emulator***
- Simulation of Magic Leap: ***Application Simulator***
- If you need ***Research Mode*** on the HoloLens then you'll need to unfortunately flash the application to the Device every time because the simulator does not support Research Mode

Recording

In this section, we will see how you can record a First Person Video for each device. You will need it while making the final video of your project.

HoloLens 2

- One way to record on the HoloLens 2 is to use the voice command and say "Start Recording" while wearing the HoloLens. The video will be saved to the Videos folder of the device.

- Another way to record is to use the stream from the Windows Device Portal.

Magic Leap 2

- The Magic Leap Hub is one way to record the video stream from the Magic Leap.

Quest 3

- You can also directly record on the Quest 3 like here.

Additional Examples

On HoloLens 2

This guide has helpful information on how to get access to the RGB camera, depth sensors, grayscale cameras, and other Research Mode sensors you may need.

These are example applications that make use of Research Mode.

If you need something like Hand Tracking or something like that, check here.

On Magic Leap 2

Please check out the official API documentation for OpenXR Unity, which also provides exemplary scripts for many sensors and use-cases.

Following link can be used for Unity settings to configure sensor input settings: here

Here is the documentation from Magic Leap 2 where you can find settings and example codes to take input from devices such as voice commands, etc here.

Quest 3 Passthrough

The following steps are taken from here.

Since the Quest 3 is a VR device by design, you will be fully immersed in a virtual environment by default. To enable an AR experience, you can enable video passthrough, which streams the two front cameras to the eye displays such that the user can perceive the real environment. This mode is active in the main menu, for example.

To activate video passthrough in your Unity application, follow these steps:

1. Make sure you've completed the setup steps described in the **Creating a Unity Project** section above.

2. Under ***Edit > Project Settings > XR Plug-in Management > OpenXR***, in the OpenXR Feature Groups, make sure that the ***Meta Quest: AR Camera (Passthrough)*** feature is enabled.

3. Under ***Edit > Project Settings > Player > Other Settings***, make sure that the Graphics APIs list *Vulkan* at the first position.

4. Under ***Edit > Project Settings > Graphics***, check which Scriptable Render Pipeline Settings are selected. By default there are three different settings, *URP-HighFidelity*, *URP-Balanced*, and *URP-Performant*.

5. Locate the Universal Render Pipeline Asset by searching for `t:UniversalRenderPipelineAsset` in the *Project* window.
    a. In the ***Inspector*** window, disable the ***Terrain Holes*** and ***HDR*** options.

6. Locate the Universal Render Data Asset by searching for `t:UniversalRendererData` in the *Project* window.
    a. In the ***Inspector*** window, disable the *Post-processing* option.
    b. Set the ***Intermediate Texture*** option to ***Auto***.

7. In your scene, locate the Main Camera in the Hierarchy window (**MRTK XR Rig > Camera Offset > Main Camera**)
    a. Add an ***AR Camera Manager*** script to the main camera.
    b. Locate the ***Camera Settings Manager*** component. For the *Opaque Display*, set the Clear Mode to ***Solid Color***, then set the ***Clear Color*** to 0 for all channels (R, G, B, A).

Server-Client Application Example

This section is still under construction, but here's potentially a helpful resource for you to get started on building a Python <-> HoloLens Client Server setup.

Additional Links and Resources

In this section you'll find links to official documentation and other resources. The MRTK3 developers and the device manufacturers provide many tutorials and examples online, and many more examples have been provided by the community in the official forums and independent blogs or videos.

Also check out the official github repositories and open issues, e.g. for MRTK3:

- https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity

Take your time to look at the available resources to make your lives easier!

For Unity

- Mixed Reality Unity Guide

For HoloLens 2

- https://learn.microsoft.com/en-us/windows/mixed-reality/
- HoloLens 2 Emulator

For Magic Leap 2

- The official Magic Leap Guide for MRTK3
- Magic Leap also provides several examples for MRTK3
- MagicLeap full documentation

For Quest 3

- https://localjoost.github.io/Passthrough-transparency-with-MRTK2-and-3-on-Quest-2Pro/