MIXED REALITY EXPLORATION TOOLKIT (MRET)

RELEASE GUIDE

Version 2022.1

June 2023

Goddard Space Flight Center
Greenbelt, Maryland

Table of Contents

Overview

The Mixed Reality Exploration Toolkit (MRET) <pronounced /mer-it/> provides a toolkit for building mixed reality environments with an emphasis on science and engineering domains. MRET is designed with the end-user in mind, so the design goal is to support the creation of XR environments without requiring the use of a 3D engine such as Unity or Unreal. It is developed in C# for Unity, but the XR environments produced by MRET are represented in XML so that the user is never forced to learn how to use Unity to produce XR environments. MRET version 22.1 is compatible and tested with Unity 2020.3.47f1 LTS.

Release Notes

Version 2022.1 is a significant departure from the framework provided in previous releases. This version introduces a new XML schema that redefines how MRET is configured at runtime. More comprehensive than the previous schema definition, this new schema (version 0.9) adds significantly more capability for serializing MRET projects at runtime, which in turn provided the ability to preserve much more customization of projects.

In conjunction with the schema, the MRET framework was extensively rewritten and reorganized to provide a more robust and componentized serialization/deserialization design.

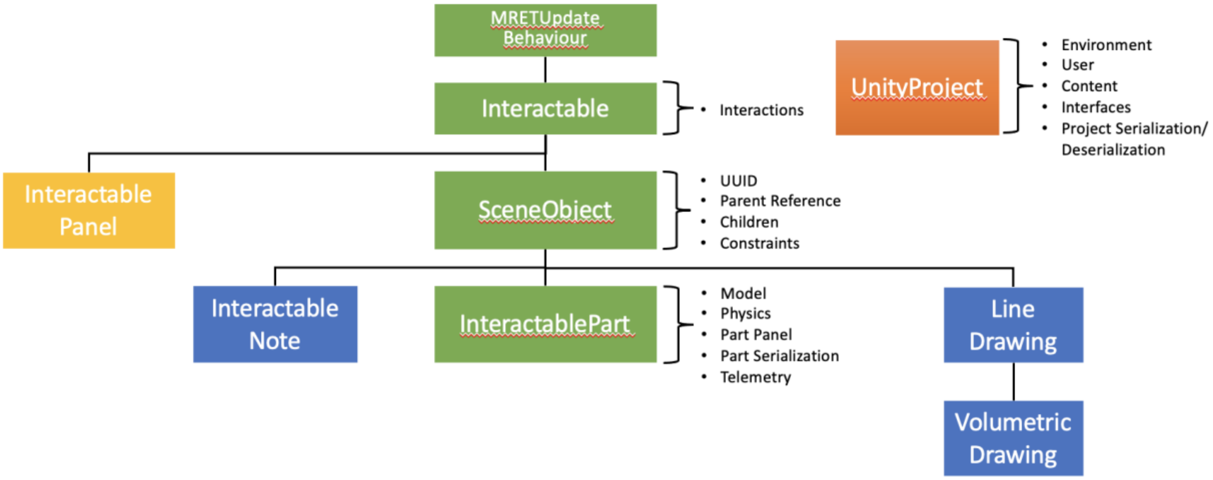


Figure 1: MRET Core Class Hierarchy (21.3.1)

Figure 1 represents the core class hierarchy as it existed in open-source release version 21.3.1. Serialization responsibilities were contained within the UnityProject. With more serialization capability in the new schema definition, the class hierarchy had to be revisited to reduce the serialization burden on a single class.

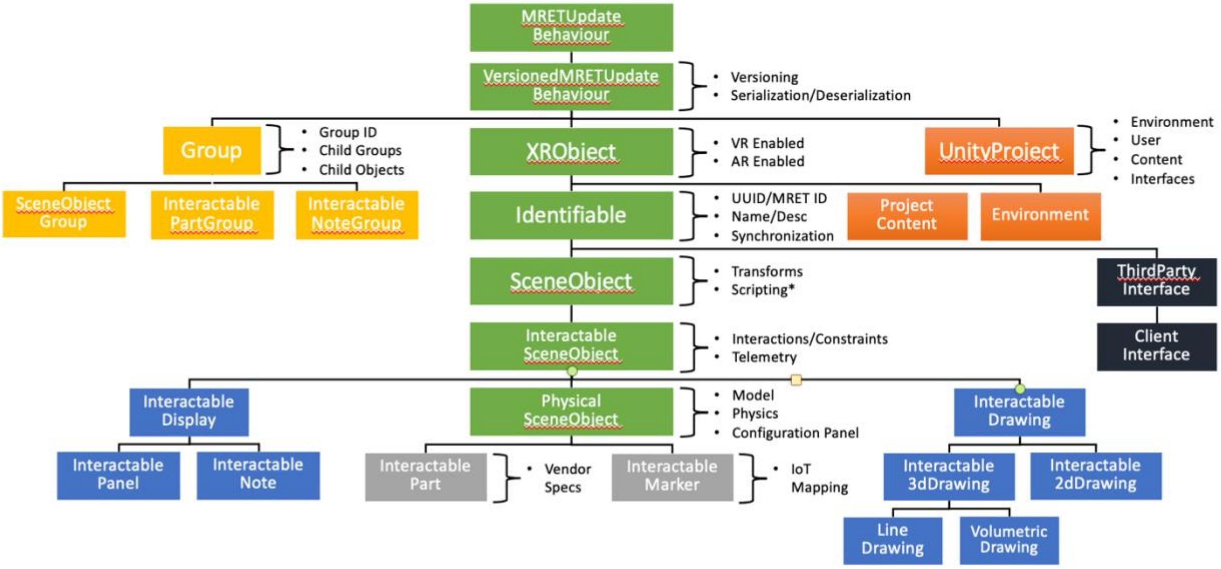


Figure 2: MRET Core Class Hierarchy (22.1)

Figure 2 represents the new MRET 22.1 core class hierarchy. The new hierarchy is more comprehensive and now includes serialization responsibilities within each class of the hierarchy.

Some additional features include:

- Live loading of CAD models from the Rapid Model Import Tool (RMIT)
- Integration with the NASA IoT Platform for telemetry
- The addition of “markers” to attach to fiducials for tracking telemetry
- Avatar enhancements
- Autosave
- Additional desktop mode features, including the ability to “select” and “grab” scene objects
- The ability to “explode” part assemblies
- 3D Webview integration
- Beta integration of dynamic terrain importing from a GeoTIFF or RAW file (not fully tested)

Setup/Configuration

Before opening MRET in Unity, you may require some configuration steps unique to your setup. The following sections address some recommended configuration steps.

Configuring for XR

MRET uses OpenXR for XR mode. Your system configuration will rely on proper configuration for your XR device brand/model. Additionally, you will have to customize the MRET project interaction profile for your setup.

Virtual Reality (VR)

SteamVR

It is recommended that you have SteamVR installed on your system. The SteamVR installer is here:

https://store.steampowered.com/app/250820/SteamVR/

Windows Mixed Reality Portal

For Windows Mixed Reality (WMR) VR devices, you will also need to install the Windows Mixed Reality Portal. You can install this via the Microsoft Store, or alternatively, you can install the portal in offline mode and WSUS here:

https://learn.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/manage-windows-mixed-reality#enable

Windows Mixed Reality Devices

If you are using a WMR VR device or HoloLens 2, you will need to add some Windows Mixed Reality Toolkit feature dependencies to the MRET project. To install them, follow these steps:

1. Download the *Mixed Reality Feature Tool* from here: https://github.com/microsoft/MixedRealityToolkit-Unity/releases
2. Point the tool to your "`<MRET>/Unity`" folder
3. Under “Mixed Reality Toolkit (0 of 10)” add: **Mixed Reality Toolkit Foundation v2.8.3**
4. Under “Platform Support (0 of 5) add: **Mixed Reality OpenXR Plugin v1.8.1**
5. sClick “Get Features”, “Import”, and “Approve”

Configuring for Automatic Asset Detection

MRET uses Unity’s Conditional Compilation Utility (CCU) to enable/disable MRET extensions (or add-on features and capabilities) to enhance the user experience in MRET[1]. These extensions require the use of third-party assets and libraries (free and/or paid) that we do not distribute with the release distribution. For MRET to properly configure itself for use in Unity, we recommend downloading and configuring the CCU prior to opening MRET in Unity for the first time. Unity’s CCU is located here:

[1] Third-party assets and libraries are not required to run MRET

https://github.com/Unity-Technologies/ConditionalCompilationUtility

if CCU is detected by Unity, then the MRET extensions will automatically be included upon detection of dependent extension classes. If the CCU is not present, then MRET will still work, but may require the user to manually configure the extensions in the MRET project. For a list of asset dependencies, please refer to the "AssetDependencies.txt" file found in the root folder of the MRET distribution.

Configuring CCU

After downloading CCU from the above link, using Windows File Explorer (or equivalent) to move the entire folder to the Assets folder in the MRET release directory. We recommend placing the contents here:

```
<MRET>/Unity/Assets/AssetsandLibraries/Non-Distributable/UnityCCU
```

Once you have the CCU in place, you may safely open MRET in Unity.

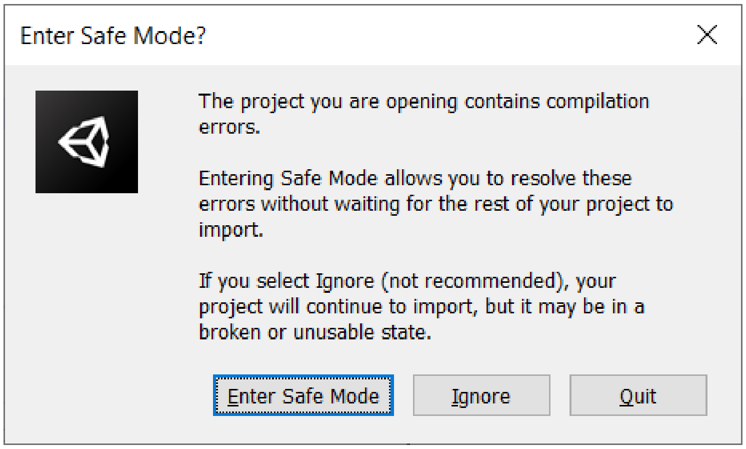
Opening MRET in Unity

When opening MRET in Unity for the first time, you will likely encounter various errors and/or dialog boxes that appear. Here are some that you may encounter and the recommended steps:

Mixed Reality Prefab Errors

If you installed the MRTK feature dependencies during the configuration process, MRTK may display prefab errors in the console. We will address the reasons for the errors in the next section and these should disappear once we have finished configuring MRET.

Enter Safe Mode


Enter Safe Mode?

The project you are opening contains compilation errors.

Entering Safe Mode allows you to resolve these errors without waiting for the rest of your project to import.

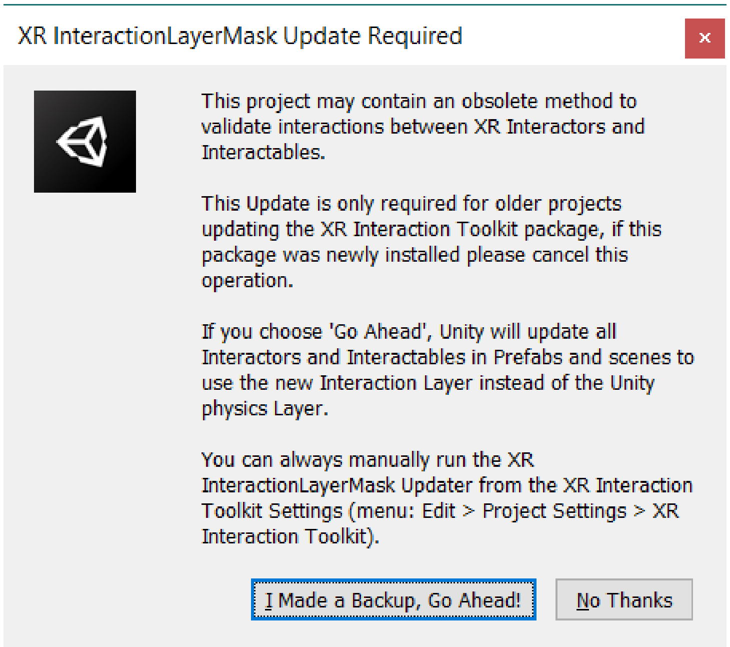
If you select Ignore (not recommended), your project will continue to import, but it may be in a broken or unusable state.

Figure 3: Enter Safe Mode Dialog

If you encounter the Enter Safe Mode dialog as shown in Figure 3, this is likely due to some MRET dependency issues. We recommend choosing "Ignore" and examining the compiler errors in the console to isolate the problem. If the issue appears to be related to a missing dependency, go to the "Edit->Project Settings…" and select "Player". Scroll down to the "Script

Define Symbols" and remove all MRET extension define symbols "MRET_EXTENSION_*", "Apply", and restart Unity.

XR InteractionLayerMask



XR InteractionLayerMask Update Required

This project may contain an obsolete method to validate interactions between XR Interactors and Interactables.

This Update is only required for older projects updating the XR Interaction Toolkit package, if this package was newly installed please cancel this operation.

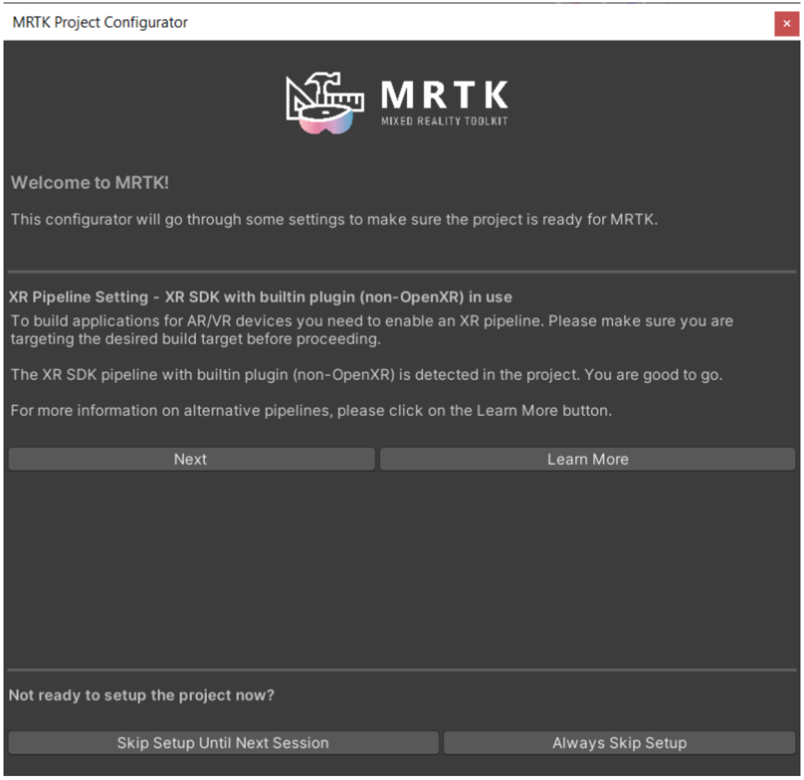
If you choose 'Go Ahead', Unity will update all Interactors and Interactables in Prefabs and scenes to use the new Interaction Layer instead of the Unity physics Layer.

You can always manually run the XR InteractionLayerMask Updater from the XR Interaction Toolkit Settings (menu: Edit > Project Settings > XR Interaction Toolkit).

Figure 4: XR InteractionLayerMask Update Required Dialog

If you encounter the XR InteractionLayerMask dialog shown in Figure 4, select "No Thanks".

MRTK Project Configurator



Welcome to MRTK!

This configurator will go through some settings to make sure the project is ready for MRTK.

XR Pipeline Setting - XR SDK with builtin plugin (non-OpenXR) in use

To build applications for AR/VR devices you need to enable an XR pipeline. Please make sure you are targeting the desired build target before proceeding.

The XR SDK pipeline with builtin plugin (non-OpenXR) is detected in the project. You are good to go.

For more information on alternative pipelines, please click on the Learn More button.

Not ready to setup the project now?

Figure 5: MRTK Project Configurator Dialog

If you installed the MRTK dependencies in the previous section, you will likely encounter the dialog in Figure 5. Select “Always Skip Setup” because the MRET scenes that use MRTK are already configured properly, and we will be configuring the interaction profiles manually in the next section.

Configuring MRET

To complete your MRET configuration for XR, you will need to select your OpenXR interaction profile. To select your correct profile, follow these steps:

1) Under “Edit->Project Settings…”, select “XR Plug-in Management.”
2) Under “Plug-in Providers”, select the “OpenXR” checkbox (ignore the warning for now)
    a. For Windows Mixed Reality devices, also select the “Windows Mixed Reality feature group” checkbox.
3) In Project settings, select “OpenXR” under the “XR Plug-in Management.”
4) Under “Interaction Profiles”, select the profile(s) that match your device.
5) Under OpenXR Feature Groups”, for WMR devices, select the “Windows Mixed Reality” checkbox and enable the features you desire.
6) In Project settings, select “Project Validation”. There shouldn’t be any warnings at this point, but if there is, select “Fix” to repair any issues.

You are now ready to run MRET.

Running MRET

MRET supports different user interaction modes: Desktop, VR, and AR. Each mode defines a rig prefab that is pulled into a Unity scene. The MRET distribution comes with three main scenes: Desktop, VR, and HoloLens 2[2]. The scenes are in this location:

```
      <MRET>Unity/Assets/MRET/Scenes
```

In the Unity editor, open the scene corresponding to your preferred interaction mode and press the play button in the editor. The VR and Desktop scenes should run without issue, but monitor the console for errors.

Possible Issues

VR Device not Detected

When running MRET, the VR headset and controllers are unresponsive. We have encountered this situation when the correct system OpenXR Runtime is not set to the proper setting for your headset. There are different configurations depending on when you are running in the Unity editor or as a standalone build.

Unity Editor

Under “Edit->Project Settings…->XR Plug-in Management->OpenXR->Play Mode OpenXR Runtime”, try explicitly setting the OpenXR Runtime to the setting for your VR device, i.e., “SteamVR” or “Windows Mixed Reality”.

Standalone Build

In SteamVR, select “Settings->Developer” and set the “Current OpenXR Runtime” to the setting for your VR device, i.e., “SteamVR” or “Windows Mixed Reality”.

Prefabs

After loading MRET in Unity for the first time, some prefabs may report errors, such as the CanvasWebViewPrefab and the WebViewPrefab (Web Browser prefabs). These prefabs are variants of the Vuplex 3D WebView distribution, so use of these prefabs will require adding the 3D WebView asset to the project.

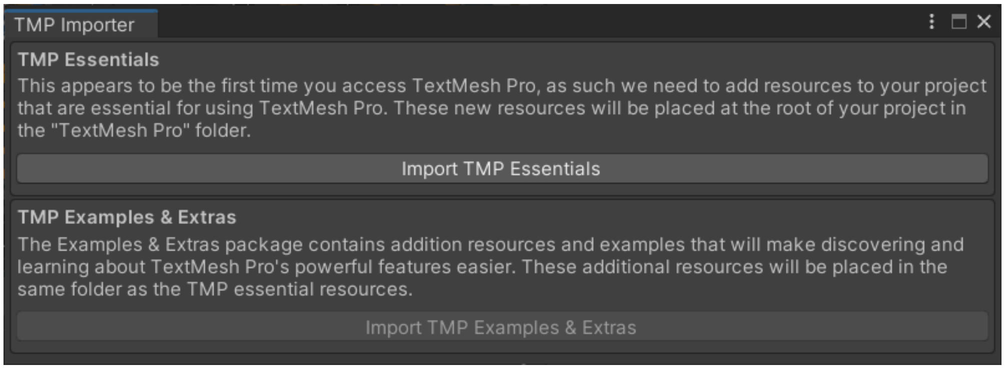
Avatars

The Avatar System is unavailable for this release due to distribution licensing for the models. The mechanism for loading the avatars will be revisited for inclusion in future releases.

Text Mesh Pro (TMP)

When loading projects for the first time, you may encounter the ‘TMP Importer’ dialog in Figure 6. These are free add-ons for Text Mesh Pro (TMP).

[2] *The HoloLens 2 scene required MRTK and must be compiled with the UWP/XAML/IL2CPP scripting backend for ARM64. Some features that use Windows plug-in DLLs will be unavailable for the HoloLens 2.*



TMP Importer

TMP Essentials

This appears to be the first time you access TextMesh Pro, as such we need to add resources to your project that are essential for using TextMesh Pro. These new resources will be placed at the root of your project in the "TextMesh Pro" folder.

Import TMP Essentials

TMP Examples & Extras

The Examples & Extras package contains addition resources and examples that will make discovering and learning about TextMesh Pro's powerful features easier. These additional resources will be placed in the same folder as the TMP essential resources.

Import TMP Examples & Extras

Figure 6: TMP Importer Dialog

We recommend importing TMP essentials.

JPL SPICE Toolkit

When loading the Lunar scenes in MRET, you may encounter the following error in the console:

An issue was encountered initializing the SpiceToolkit: SPICE(NOSUCHFILE)

MRET uses the JPL SPICE Toolkit to add accurate positioning of the Sun and the Earth in the Apollo and Lunar South Pole scenes. The SPICE Toolkit integration in MRET is limited to a subset of SPICE Toolkit functions and was done to demonstrate lighting and shadows on the Moon surface. For the calculations to work in the lunar scenes, the SPICE Toolkit requires SPICE data, or kernels, to perform the calculations. For more information on SPICE, please refer to this SPICE overview:

https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/03_spice_overview.pdf

The MRET distribution does not include the SPICE kernels required for the calculations. They are maintained at Navigation and Ancillary Information Facility (NAIF) [https://naif.jpl.nasa.gov/naif/]. To use the SPICE Toolkit capability in MRET, please download the following kernels:

- naif0012.tls
- de421.bsp
- pck00019.tpc
- moon_080317.tf
- moon_pa_de421_1900-2050.bpc
- earth_fixed.tf
- earth_070425_370426_predict.bpc

The Moon meta kernel file used by MRET is included with the distribution and is already configured to point to the list of required kernels. The meta kernel file resides in this location:

```
<MRET>Unity/Assets/Content/SPICE/moon_kernel.tm
```

To use the SPICE kernels, place them in the same folder as the meta kernel. CAUTION: You may edit the meta kernel file to point to a different location for each of the kernels, but the meta kernel path length is limited, so we recommend keeping the paths relatively short.

Once the kernels are in place, the SPICE Toolkit errors should disappear, and the SPICE calculations will begin once you enter the lunar scenes in MRET.

Missing Behavior Warnings

You may encounter warnings related to missing behaviors in game objects. These are all warnings that a third party asset script referenced in a prefab is unavailable. When these scripts are not available, MRET will still run, but some of the prefabs will not behave as intended. To rectify the warning, install the associated asset referenced in the AssetDependencies.txt file distributed with this MRET release. Some examples are:

RockVR

```
The referenced script on this Behaviour (Game Object ‘SpectatorCamera’) is missing!
The referenced script on this Behaviour (Game Object ‘VideoCamera’) is missing!
```

Digital Ruby Free Earth Planet

```
The referenced script on this Behaviour (Game Object ‘EarthMain’) is missing!
The referenced script on this Behaviour (Game Object ‘EarthClouds’) is missing!
The referenced script on this Behaviour (Game Object ‘SunObject’) is missing!
```

References/Notes