How to use the Multi-AR Examples package

Introduction

Augmented reality (AR) comes on several different platforms these days. The biggest ones, hardware and OS-specific, are Apple AR-Kit for iOS, Google AR-Core for Android and MS Windows Mixed Reality. The goal of the Multi-AR package is to provide an easy way to deal with the specifics of different AR platforms, and separate it from the development of the AR scenes. Currently the Multi-AR Examples-package supports AR-Kit on iOS, AR-Core on Android, HoloLens & Windows Mixed Reality on UWP, and Meta-2 AR on Windows Standalone. Support for more AR platforms may be added, as they become available.

Setup and running

The setup of Multi-AR Examples is straightforward:

1. Download and import the Multi-AR package into new Unity project.
2. Please, don't import any external AR platform-specific packages.
3. Open a demo-scene from the MultiAR/DemoScenes-folder.
4. Build and run the scene on the target platform – iOS for AR-Kit, Android for AR-Core, UWP (Universal Windows Platform) for Windows MR, or Windows standalone for Meta-2.
5. Look at the settings of the MultiARManager-component in the scene. Try to change some of them and re-run the scene on the target platform, to see the effect.
6. For AR-Kit setup look at 'Setup of Apple AR-Kit'-section below.
7. For AR-Core setup look at 'Setup of Google AR-Core'-section below.
8. For Windows-MR setup look at 'Setup of HoloLens and Windows-MR'-section below.
9. For Meta-2 setup look at 'Setup of Meta-2 AR'-section below.

The core components

Multi-AR Examples consists of one main component called MultiARManager, interface-components for each of the supported AR platforms and several supporting components. The core components reside in the MultiAR/CoreScripts-folder. MultiARManager and the available platform interfaces are components of the MultiARController-game object in the demo scenes. For your convenience, the MultiARController game object and all its components are stored as prefab with the same name in the MultiAR/CoreScripts/Prefabs-folder.

How to reuse the 'Multi-AR Examples'-components in your Unity projects

To reuse the 'Multi-AR Examples' components in your own Unity project, do as follows:

1. Import the Multi-AR package into your project, as described above. Then delete the MultiAR/DemoScenes-folder. Alternatively, copy the MultiAR/CoreScripts-folder and the platform-specific folders (GoogleARCore, UnityARKitPlugin, etc.) from this project to your project.
2. Drag the MultiARController-prefab from the MultiAR/CoreScripts/Prefabs-folder to your scene.
3. Check if the settings of the MultiARManager-component are correct for your use case.
4. That's all! You are ready to build scene and run it on the supported AR-platforms.
5. Look at the public API of MultiARManager. You may invoked any mothod by calling 'MultiARManager.Instance.xxxx()'. Look at the demo scenes, if you need examples.

Setup of Apple AR-Kit

To run your project on iOS with AR-Kit, follow these steps and requirements:

1. The AR-Kit plugin requires Unity 2017.1.0 or later.
2. It requires iOS 11.3 or later installed on the target device.
3. It requires XCode 9.3 or later, with iOS SDK that includes ARKit Framework.
4. It requires iOS device that supports ARKit (i.e. iPhone 6S or later, iPad 2017 or later).
5. To build for iOS open the Build settings, select 'iOS' as Platform and press 'Switch Platform'.
6. Press 'Player settings' and under 'Other Settings' set 'Bundle identifier' for your product, 'Camera usage description' to a meaningful message, and optionally 'Location usage description', as well.
7. The latest release of the AR-Kit plugin may be found in the Unity asset store.

Setup of Google AR-Core

To run your AR project on Android with AR-Core, follow these steps and requirements:

1. The AR-Core plugin requires Unity 2017.3.0 or later.
2. To build for Android devices, you need to have the Android Studio installed. Here is the download link: https://developer.android.com/studio AR-Core requires Android SDK version 7.0 Nougat (API Level 24 or higher). More information can be found here: https://developers.google.com/ar/develop/unity/getting-started#setting_up_your_development_environment
3. Make sure your Android device supports AR-Core. Here is the list of supported devices: https://developers.google.com/ar/discover/supported-devices
4. Prepare your device: Enable 'Developer options' & 'USB debugging'. AR-Core will be installed automatically, when you run your AR-Core enabled application for a first time.
5. To build for Android open the Build settings, select 'Android' as Platform and press 'Switch Platform'.
6. Press 'Player settings' and under 'Other Settings' disable 'Multithreaded rendering', set the 'Package name' as needed, 'Minimum API Level' to 'Android 7.0 Nougat (API level 24)'. Finally, under 'XR Settings' enable 'ARCore supported'. Then you are ready to build. More information can be found here: https://developers.google.com/ar/develop/unity/quickstart-android#configure_build_settings
7. More information on how to get started with AR-Core can be found here: https://developers.google.com/ar/develop/unity/getting-started
8. The latest release of the Google's AR-Core Unity plugin may be found in its Git repository: https://github.com/google-ar/arcore-unity-sdk.git

Setup of HoloLens and Windows-MR headsets

To run your AR project on HoloLens or Windows-MR headsets in Unity, follow these steps and requirements:

1. For HoloLens and Windows-MR headsets development use Unity 2017.3.0 or later.
2. For HoloLens and Windows-MR headsets builds and app deployment use VS-2017 or later.
3. Open the Build settings in Unity, select 'Universal Windows Platform' and press 'Switch Platform'. Make sure .NET is selected as scripting backed in 'Other settings'. In 'Publishing settings' enable the following capabilities: Spatial Perception, and optionally Webcam and/or Microphone.
4. *HoloLens only*: Before running the scene, open 'Window / Holographic Emulation', select 'Remote to Device' as emulation mode, set the IP address of the device (start the Remoting Player on the device), and then press 'Connect' to establish connection between the Unity editor and HoloLens.
5. *WinMR headsets only*: Before running Unity editor, start the 'Windows Mixed Reality' app.
6. Run the scene in Unity editor. You can see it right away on the HoloLens or WinMR headset.
7. To deploy the scene to the device, build the Unity scene, then open the resulting C# project in Visual Studio, build it in VS, and run it on the device (HoloLens) or local machine (WinMR headset).
8. More information on Windows-MR Unity development can be found here: https://developer.microsoft.com/en-us/windows/mixed-reality/unity_development_overview

Setup of Meta-2 AR

To run your AR project on Meta-2, follow these steps and requirements:

1. For Meta-2 development, please use Unity 2017.3.0 or later.
2. Install Meta SDK. Here is the download link: http://devcenter.metavision.com/getstarted
3. Import the optional 'Meta2Interface.unitypackage'.
4. Open the Build settings in Unity, select 'PC, Mac & Linux Standalone', target platform 'Windows' and press 'Switch Platform'.
5. Then you can run the scene in Unity editor or create a standalone build.

Demo Scenes

The demo scenes in the package are located in the MultiAR/DemoScenes-folder. The short descriptions of all demo scenes are available in the online documentation.

More Information, Support and Feedback

Online Documentation: https://ratemt.com/mardocs/
Web: http://rfilkov.com

Contact: http://rfilkov.com/about/#contact
Twitter: https://twitter.com/roumenf