AR Installation & Setup Guide

This guide will walk you through how to install and setup Unity for mobile AR development. The guide may look daunting and complicated, but it is mainly the setup for your mobile device that adds a lot of steps.

Setup GitLab

First, it is highly recommended to setup a GitLab repository to keep track of your project and it could save you at times. To do this you can

1. Navigate to https://gitlab.au.dk and login with your AU credentials.
   a. Where one group member clicks "New project"
   b. Then give the repository a name (e.g., your team-name)
   c. Then click "Create project"
2. Then invite your group members to the project. This can be found under "Project information" -> "Members".

Installing Unity

The next step is to download and install the Unity Hub and Unity Editor. During this process, you will need to select a Unity plan and create a Unity ID if you don't have these already.

We have made a general walk through on YouTube, which shows you the installment process. But please use the information below, such that you use the right versions, folders, and so on.

Installing Unity Hub + Unity (Video Guide)

1. Download the Unity hub. When you have installed Unity Hub, you can acquaint yourself with Unity Hub by following this tutorial.
2. To install Unity, do the following in Unity Hub:
   a. Click: "Installs" (left menu)
   b. Click: "Install Editor"
   c. Select Unity version: **2020.3.*xx*** LTS (where *xx* is not that important, so just pick the newest version. For ease of use, please select the same *xx* within the group) and click: "Install".
   d. You are now asked to "Add modules". You should install the following:

        i. Platforms, choose either Android or iOS (since this AR walkthrough will focus on mobile development):
            1. Android Build Support or
            2. iOS Build Support
        ii. Documentation, needed to get documentation on functions and such.
        iii. You may also want to install Visual Studio if you do not have any plans for an IDE (development tool) to use. You can alternatively use VS Code or JetBrains Rider.
    e. Click "Continue" and "Install" until it starts downloading and installing.

Create a new Unity project and push it to git

This happens in two stages. To begin with, one of your team **HAVE TO** do some things on only their compute (probably the person who created the repository). Then the other team members perform the last step.

First team member:

1. Clone your GitLab repository you created in the beginning to your computer. You should now have a folder somewhere (e.g., "~/[YOUR REPO NAME]").
2. Open Unity Hub and create a new **3D Core** project.
    a. Set the Unity "Project name" to something (e.g., "FunAR") and set "Location" to your git repository. This should create a new folder in the git repository folder containing the Unity project.
3. Copy the attached ".gitignore" file for Unity into your git repository. This ".gitignore" will ignore all the generated files that Unity will reproduce without a problem. Git only needs the files that are not generated.
4. Commit and push everything to git.
5. Your folder structure on https://gitlab.au.dk should look like this (which should differ from your OS folder due to the ".gitignore"):
    a. Root
        i. [Unity Project Name]
            1. Assets
            2. Packages
            3. ProjectSettings
        ii. .gitignore
        iii. README.md

Other team members:

1. Wait for your first team member to push and verify that everything went well
2. Clone your GitLab repository
3. Open Unity Hub and open your project (e.g., "~/[Your Team Name]/[Your Unity Project Name]")

Optionally, install a code editor

You may want to have Unity work with your code editor of choice, here we provide two ways of installing two popular choices.

Installing VS Code and having it work with Unity

When you install a Unity version with default settings, Visual Studio will also be installed. However, it is possible to have VS Code work with Unity including IntelliSense and code completion. To achieve this, follow the guide by Microsoft here.

Installing JetBrains Rider and having it work in Unity

Alternatively, if you like the IDEs from JetBrains, you can also use Rider.

1. Install JetBrains Toolbox App
2. Create a JetBrains Account (free as a student)
3. Install Rider directly from JetBrains Toolbox
4. Start using Rider with your Unity project

Installing & setting up AR Foundation

Since this AR walkthrough will focus on mobile AR, you will need to install some additional packages inside your Unity project. This page will guide you through the first steps, so give it a quick read.

5. Install *AR Foundation* (at least version 4.1) from the Package Manager (menu: **Window > Package Manager**). Remember to add your platform-specific AR packages: ARKit XR Plugin (iOS) and/or ARCore XR Plugin (Android).
6. Create a new scene and give it a name (e.g., "ARFoundation") and add an "AR Session" and "AR session origin" to it.
7. Add plane detection to your scene by adding a "AR Plane Manager" to "AR session origin" with the plane prefab: "AR Default Plane".
8. When working with AR, you typically (always?) have one camera and one camera only in your scene. Thus, delete the default "Main Camera" from your scene.

9. Also, remember to select the "AR Camera", then in the inspector, in the dropdown with "Untagged" (just above where it says "Transform"), make sure it is tagged as "MainCamera", otherwise "Camera.main" will not work!

Building and running the project

With your scene setup and 'ready' it is now time to test it on your phone.

1. Switch the build target to iOS or Android and click "Add Open Scene" (menu: **File > Build Settings**, or **Ctrl+Shift+B on Windows**)

For Android:
To get AR Foundation to build successfully for Android you will need to change a few extra things:

1. First, with the **Build Settings** window open, click **Player Settings...** in the lower left
2. Expand the **Other Settings** dropdown
    a. Under **Rendering > Graphics APIs**, **uncheck Auto Graphics API**, click **Vulkan**, then click the minus icon in the lower right of that box. This removes **Vulkan** as it is not supported on Android
    b. Scroll down until you reach **Identification > Minimun API Level**. Change this value to **Android 7.0 (API level 24)** *(If you have an older device you will need to download some additional things...)*
    c. Then change **Target API Level** to **Android 10.0 (API level 29)** (If you have an older device you will need to download some additional things...)
    d. Now find Configuration > Scripting Backend and set that to IL2CPP
    e. Then find **Configuration** -> **Target Architectures**, untick **ARMv7** and tick **ARM64**. This is required as the current versions of ARCore no longer support 32-bit
    f. Now navigate to **XR Plug-in Management** in the left bar
    g. Make sure the Android tab is selected and tick **ARCode** under **Plug-in Providers**
    h. Then navigate to **XR Plug-in Management > ARCore** in the left bar
    i. Change the dropdown for **Depth** to **Optional**
3. You can now close the **Player Settings** window and go back to the **Build Settings** window.
4. If you have yet to enable **USB Debugging** on your Android phone, do the following:
    a. Open Settings > Find About Phone
    b. Click **Build Number** 5 or more times until it says **Developer Options** is enabled

   c. Navigate to **Developer Options**, usually in **System**
   d. Scroll down until you reach the **Debugging** section
   e. Make sure USB-Debugging is enabled (NOTE: You should disable this setting when not developing, as it makes your device more vulnerable...)
5. Plug your Android phone into your computer
6. Back in Unity in the **Build Settings** window, click the **Refresh** button on the right side
   a. Then select your device in the dropdown to the left of that button
   b. Click the **Build And Run** button which will prompt you for a save location of the application
   c. Create a new folder called **Builds** and open it
   d. Then type in a name in the bottom field and click **Save**
   e. Unity will now build your application and it should appear on your phone

For iOS:
First of all, you need to have a computer running Macintosh as you need to install Xcode! To get AR Foundation to build successfully for iOS you will need to change a few extra things:

1. Install Xcode (**App Store**: Search for Xcode and install)
2. In Unity. Go to Edit > Project Settings
   a. > XR Plug-in Management: tick ARKit
   b. **> Player Settings > iOS tab** (should be default if your target platform is iOS)
   c. Select a meaningful and kind of unique **Company name** and **Product Name** (it will create your bundle identifier; which will be com.CompanyName.ProductName which has to be unique to any other app in the world)
   d. Other Settings > Tick Requires ARKit Support
   e. Other Settings > Architecture: ARM64 (might be default)
   f. Other Settings -> Target minimun iOS Version: 11.0
3. In Unity. Build and run your project (**File > Build Settings > Build and Run**)
   a. Create a "Build" folder in your project (e.g. `~/[Your PROJECT NAME]/AR22/Build`) to hold your project build files
4. Open project in Xcode
   a. Xcode > Preferences > Accounts
      i. Add your AppleID by clicking **+**

        ii. Click your AppleID -> **Manage Certificates**: add your laptop
    b. Connect your iPhone to your laptop
        i. Change **Any iOS Device** to your iPhone
        ii. Click Unity-iPhone project > Signing & Capabilities: click Automatically manage signing, and select Team to [your name] (personal team)
        iii. Click play
5. On your iPhone: Settings > General > VPN & Administration: allow your app

Hopefully, at this point your app should be running on your phone.