

EXPERIMENT NO 7

Getting started with AR Foundation

What's next

- Get started with the optional [ARCore Extensions](#) package, which provides additional ARCore functionality for the AR Foundation framework.
- [Create an AR game using Unity's AR Foundation](#) (codelab).

Overview of ARCore and supported development environments

bookmark_border

ARCore is Google's platform for building augmented reality experiences. Using different APIs, ARCore enables your phone to sense its environment, understand the world and interact with information. Some of the APIs are available across Android and iOS to enable shared AR experiences.

ARCore uses three key capabilities to integrate virtual content with the real world as seen through your phone's camera:

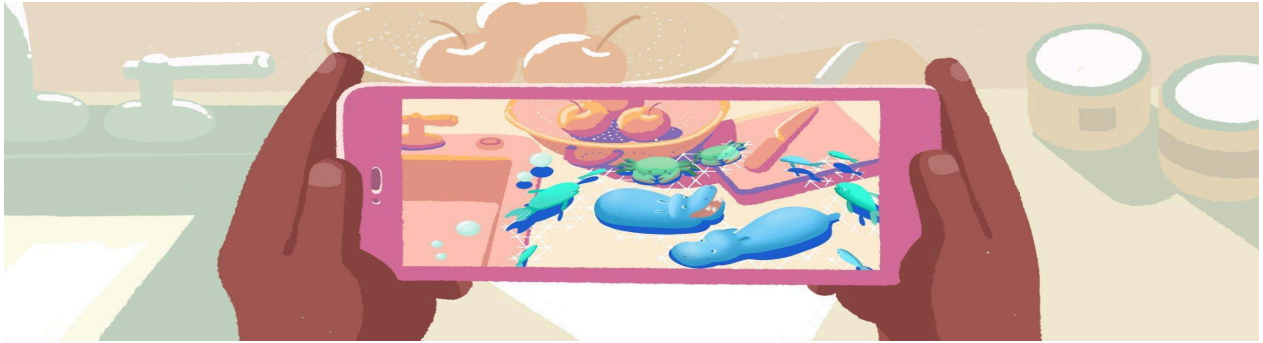
- [Motion tracking](#) allows the phone to understand and track its position relative to the world.
- [Environmental understanding](#) allows the phone to detect the size and location of all type of surfaces: horizontal, vertical and angled surfaces like the ground, a coffee table or walls.
- [Light estimation](#) allows the phone to estimate the environment's current lighting conditions.

Supported devices

ARCore is designed to work on a wide variety of qualified Android phones running Android 7.0 (Nougat) and later. A full list of all supported devices [is available here](#).

How does ARCore work?

EXPERIMENT NO 7



Fundamentally, ARCore is doing two things: tracking the position of the mobile device as it moves, and building its own understanding of the real world.

ARCore's motion tracking technology uses the phone's camera to identify interesting points, called features, and tracks how those points move over time. With a combination of the movement of these points and readings from the phone's inertial sensors, ARCore determines both the position and orientation of the phone as it moves through space.

In addition to identifying key points, ARCore can detect flat surfaces, like a table or the floor, and can also estimate the average lighting in the area around it. These capabilities combine to enable ARCore to build its own understanding of the world around it.

ARCore's understanding of the real world lets you place objects, annotations, or other information in a way that integrates seamlessly with the real world. You can place a napping kitten on the corner of your coffee table, or annotate a painting with biographical information about the artist. Motion tracking means that you can move around and view these objects from any angle, and even if you turn around and leave the room, when you come back, the kitten or annotation will be right where you left it.

For a more detailed breakdown of how ARCore works, check out [fundamental concepts](#).

ARCore provides SDKs for many of the most popular development environments. These SDKs provide native APIs for all of the essential AR features like motion tracking, environmental understanding, and light estimation. With these capabilities you can build entirely new AR experiences or enhance existing apps with AR features.

Requirements

Depending on your target platform, the **AR Foundation** package requires the following:

[Android](#)
[iOS](#)

Hardware

- An [ARCore supported](#) Android device or emulator

EXPERIMENT NO 7

- A USB cable for connecting your device to your development machine

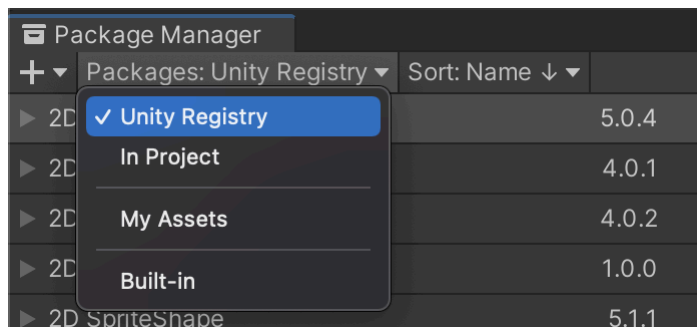
Software

- [Unity 2019.4.3f1 or later](#) with [Android Build Support](#)

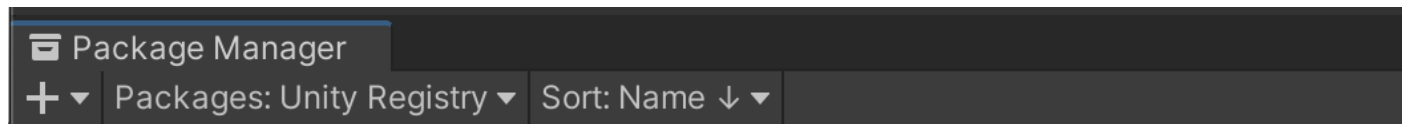
Install AR Foundation

Follow these steps to install the **AR Foundation** Unity package.

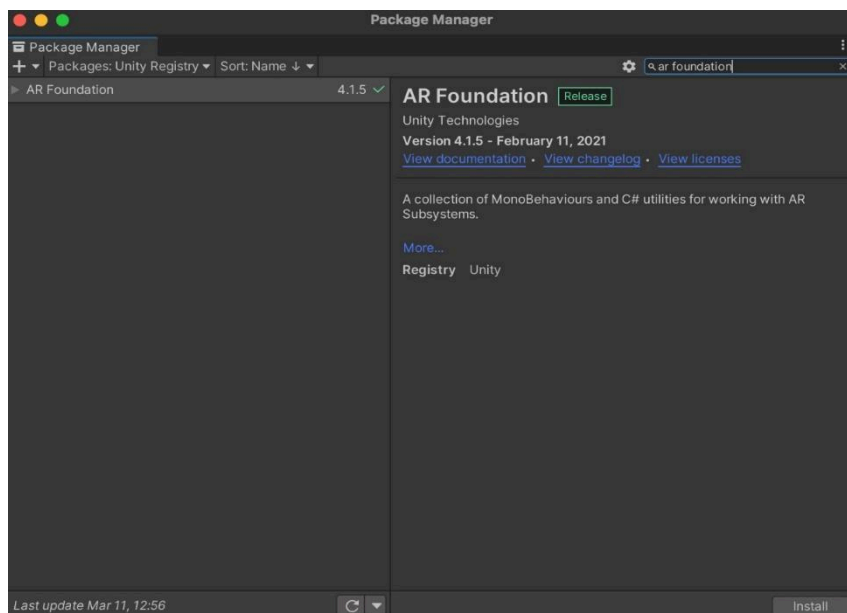
1. Open an existing Unity project, or create a new 3D project.
2. Navigate to **Window > Package Manager**.
3. Next to **Packages**, select **Unity Registry**.



4. In the search bar, type "AR Foundation".



5. Click **Install**.



EXPERIMENT NO 7

6. Configure the Render Pipeline

[AR Foundation 4.x](#) [AR Foundation 5.x](#) [AR Foundation 6.x](#)

Note: Both the [Built-in Render Pipeline](#) and the [Universal Render Pipeline](#) are compatible with the AR Foundation package, but the latter requires [additional steps](#) to configure.

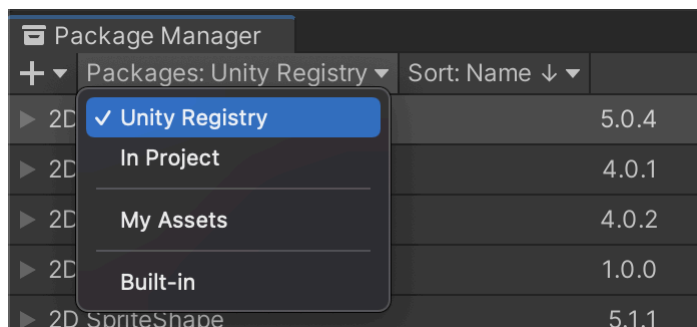
Install and enable platform-specific plugin packages

The AR Foundation package presents an interface for Unity developers to use, but doesn't implement any AR features itself. To use AR Foundation on a target device, you'll also need to install separate packages and enable the corresponding plugin for each respective platform.

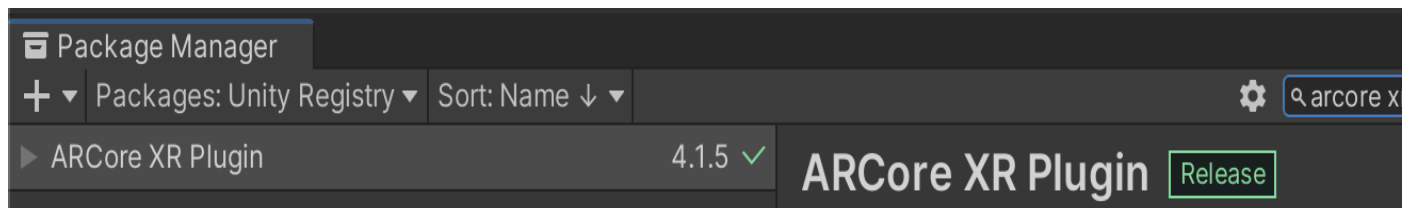
[Android](#) [iOS](#)

The [ARCore XR Plugin](#) package lets you build augmented reality apps for Android devices. Follow these steps to install it in your Unity project if your application targets Android devices. Use the same version as the *AR Foundation* package to avoid any compatibility issues.

1. In your project, navigate to **Window > Package Manager**.
2. Next to **Packages**, select **Unity Registry**.

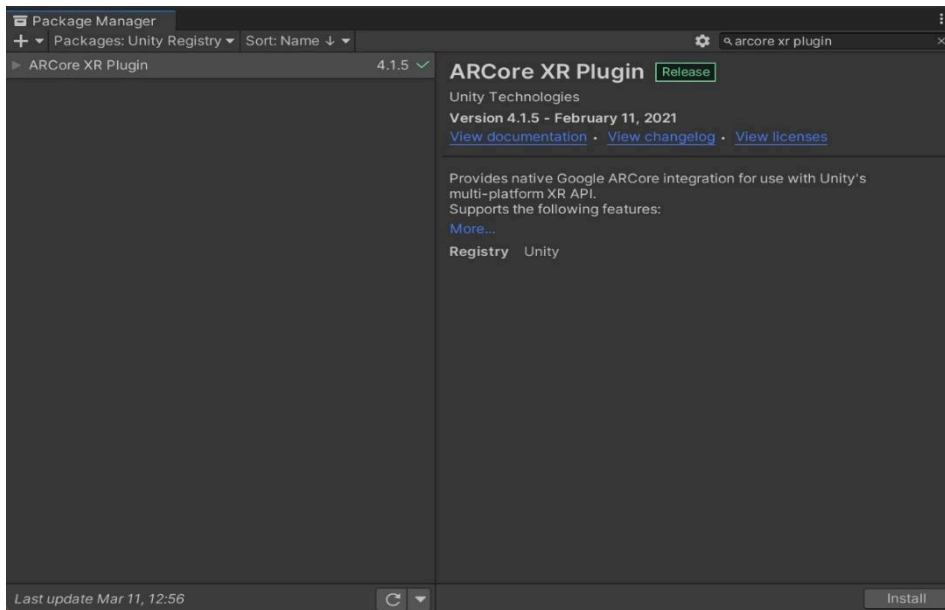


3. In the search bar, type "ARCore XR plugin".

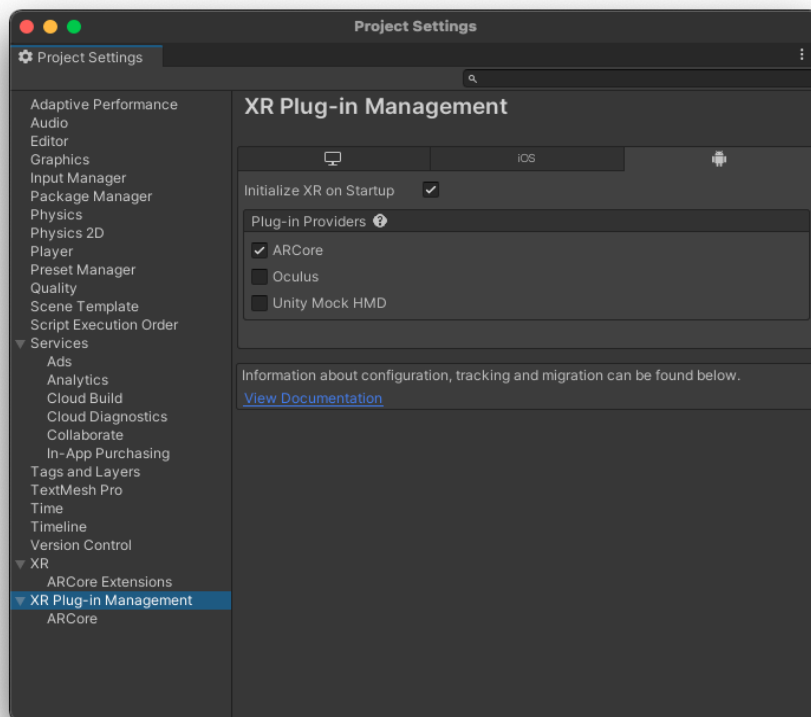


4. Click **Install**.

EXPERIMENT NO 7



5. Navigate to **Edit > Project Settings**. In **XR Plug-in Management**, open the **Android** tab and enable **ARCore**.



Configure an AR Session and add AR Foundation components to your scene

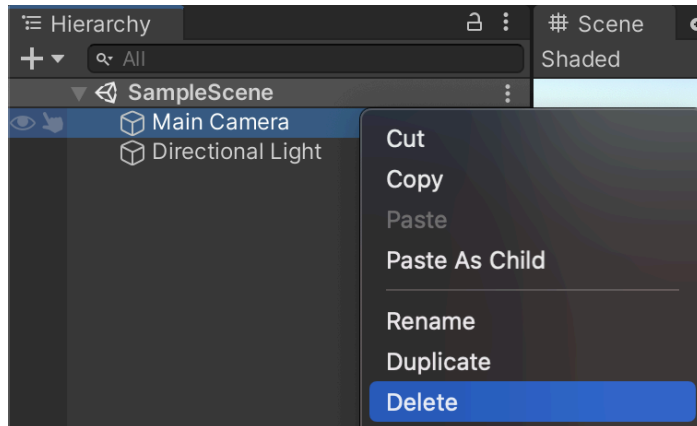
EXPERIMENT NO 7

A scene needs an AR session to enable [AR processes](#), such as motion tracking, environmental understanding, and lighting estimation. You will need the following game objects to support an AR session:

[AR Foundation 4.x](#) [AR Foundation 5.x](#) [AR Foundation 6.x](#)

- **AR Session**: Controls the lifecycle of an AR experience.
- **AR Session Origin**: Transforms AR coordinates into Unity world coordinates.

Before adding the new game objects, delete the default **Main Camera**. It will be replaced by a new **AR Camera** in the **AR Session Origin**.



Add the new AR game objects to your scene: right-click the **Hierarchy** pane and select **XR**. Add a new **AR Session** and a new **AR Session Origin** game object.

Configure Player Settings

[Android](#) [iOS](#)

1. Navigate to **File > Build Settings** to open the [Build Settings](#) window.
2. Under **Platform**, choose **Android** and click **Switch Platform**.
3. Click **Player Settings**.
4. Apply the following changes:

Player Settings > ...	Value
Other Settings > Rendering	Uncheck Auto Graphics API. If Vulkan is listed under Graphics APIs, remove it as Vulkan is not yet supported by ARCore.
Other Settings > Package Name	Create a unique app ID using a Java package name format. For example, use <code>com.example.helloAR</code> .

EXPERIMENT NO 7

Other Settings > Minimum API Level	If you are building an AR Required app, specify Android 7.0 'Nougat' (API Level 24) or higher. If you are building an AR Optional app, specify Android API Level 19 or higher.
Other Settings > Scripting Backend	Select IL2CPP instead of Mono to allow ARM64 support to be enabled in the next step. During development: Use Mono + 32-bit (ARMv7) Install a FAT (32-bit + 64-bit) ARCore APK When shipping to the Play Store: Use IL2CPP Enable both 32-bit (ARMv7) and 64-bit (ARM64) to meet Play Store 64-bit requirements Optional (supported in 2018.3+): In Build Settings, enable Android App Bundles
Other Settings > Target Architectures	To meet the Google Play 64-bit requirement, enable ARM64 (64-bit ARM). Leave ARMv7 (32-bit ARM) enabled to support 32-bit devices.

Enable AR in your AR Foundation app (Android only)

bookmark_border

Enable AR to use augmented reality features in your new or existing app.

Configure your app to be AR Required or AR Optional

To save space on individual devices, all AR features are stored in an app called [Google Play Services for AR](#) that is updated separately by the Play Store. Android apps that use AR features communicate with Google Play Services for AR using the ARCore SDK. An app that supports AR features can be configured in two ways: **AR Required** and **AR Optional**. This designation determines how the app interacts with the Google Play Services for AR app.

An **AR Required** app cannot function without ARCore. It requires an [ARCore supported device](#) that has installed Google Play Services for AR.

- The Google Play Store will only make AR Required apps available on devices that support ARCore.

EXPERIMENT NO 7

- When users install an AR Required app, the Google Play Store will automatically install Google Play Services for AR on their device. However, your app must still [perform additional runtime checks](#) in case Google Play Services for AR is out of date or has been manually uninstalled.

An **AR Optional** app uses ARCore to enhance existing functionality. It has optional AR features which are only activated on ARCore supported devices that have installed Google Play Services for AR.

- AR Optional apps can be installed and run on devices that don't support ARCore.
- When users install an AR Optional app, the Google Play Store will **not** automatically install Google Play Services for AR on the device.

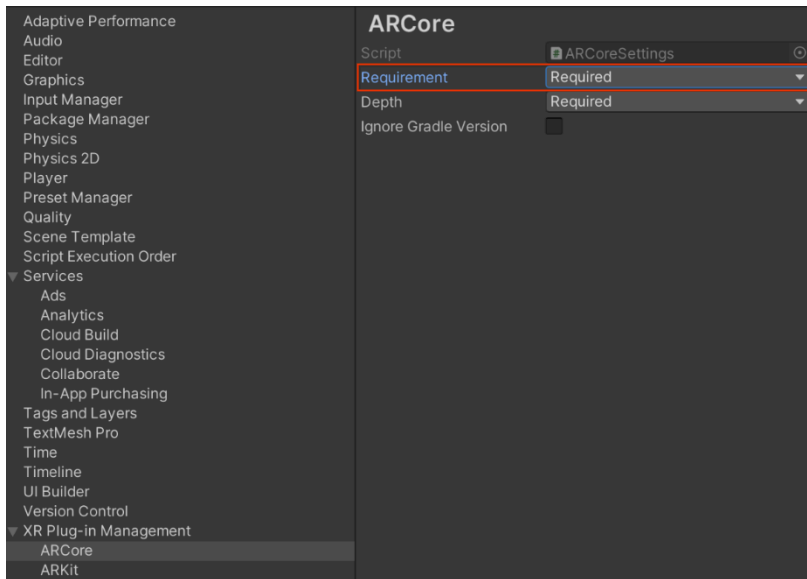
	AR Required
AR Feature usage	Your app needs ARCore for basic functionality.
Play Store visibility	Your app is only listed in the Play Store on devices that support ARCore .
Google Play Services for AR installation method	The Play Store installs Google Play Services for AR alongside your app.
Android <code>minSdkVersion</code> requirements	Android 7.0 (API Level 24)
Must use ARSession.state to check ARCore support and install status	<code>check_circle</code>
Must use the ARSession.Install() coroutine to install Google Play Services for AR	<code>check_circle</code>

Make your app AR Required

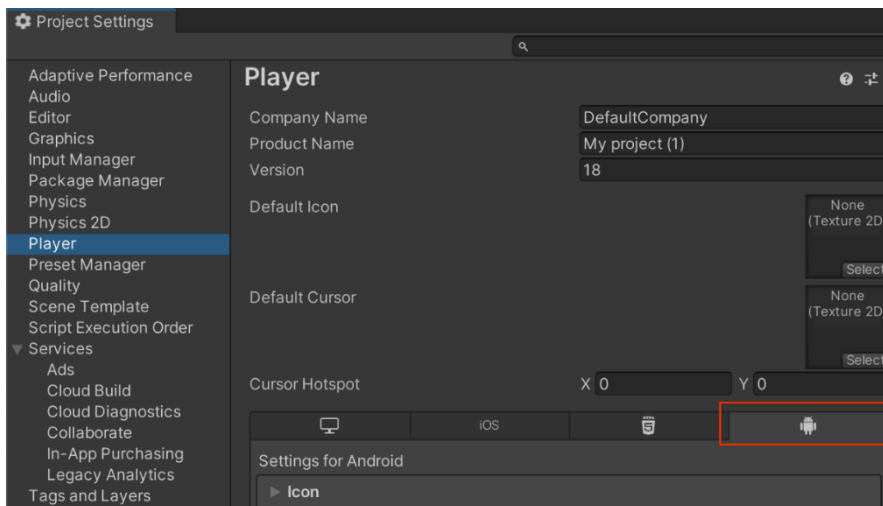
[Android](#)[iOS](#)

- Navigate to **Edit > Project Settings > XR Plug-in Management > ARCore. Requirement** is set to **Required** by default.

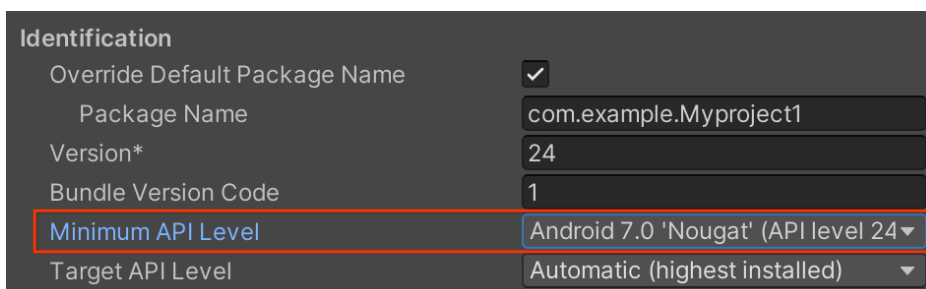
EXPERIMENT NO 7



2. Click **Edit > Project Settings > Player**, select the tab with the Android icon.



3. Under **Other Settings > Identification**, specify a **Minimum API Level** that is 24 or higher.



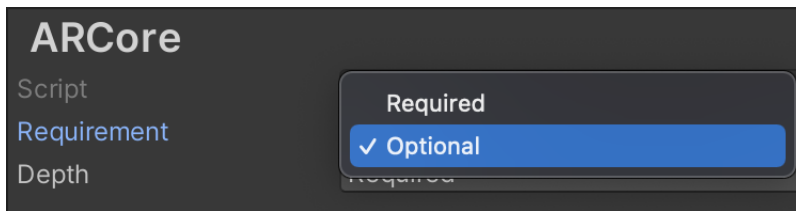
Make your app AR Optional

[Android](#)[iOS](#)

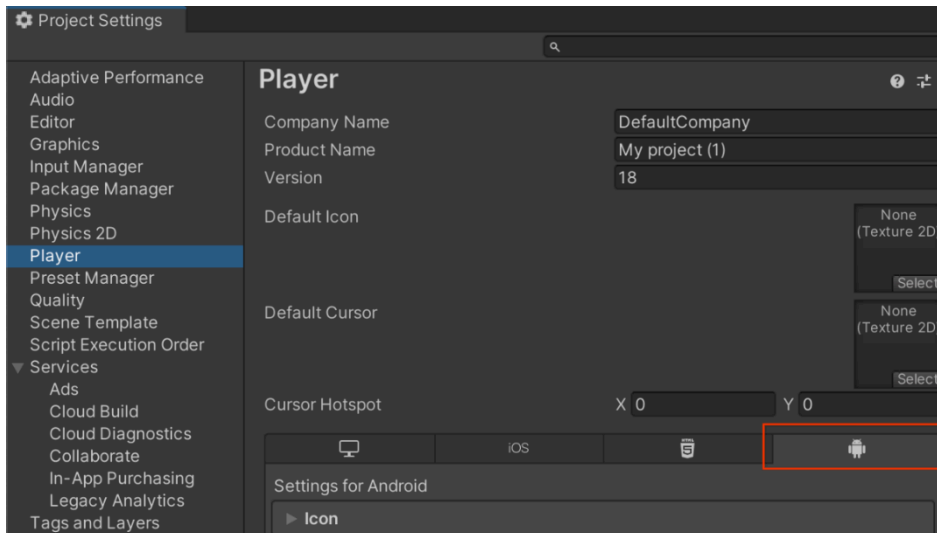
1. Navigate to **Edit > Project Settings > XR Plug-in Management > ARCore**.

EXPERIMENT NO 7

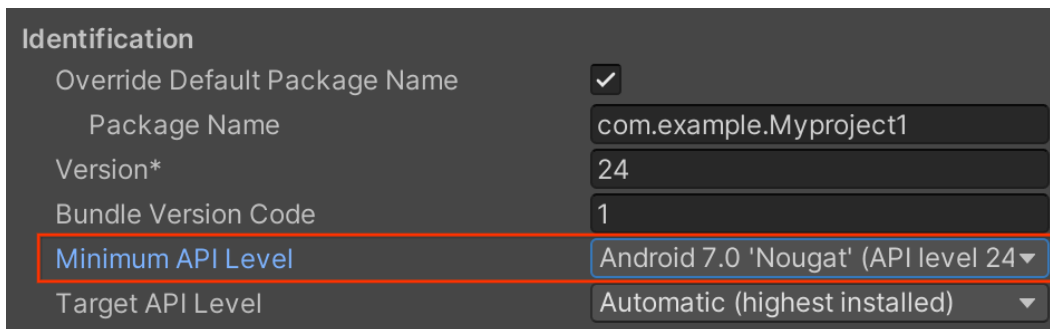
- From the **Requirement** drop-down menu, select **Optional** to set your app to AR optional.



- Click **Edit > Project Settings > Player**, select the tab with the Android icon.



- Under **Other Settings > Identification**, specify a **Minimum API Level** that is 24 or higher.



Perform runtime checks

Check if Google Play Services for AR is installed

Both AR Required and AR Optional apps must use the [ARSession.Install\(\) coroutine](#) before creating an ARCore session to check whether a compatible version of Google Play Services for AR is (still) installed and to ensure that all required ARCore device profile data has been downloaded.

EXPERIMENT NO 7

During runtime, AR Required apps will automatically perform runtime checks to ensure that the target device:

- Supports ARCore
- Has installed Google Play Services for AR — if this is not installed (possibly because it was removed by the user) or the version is too old, your app will prompt the user to install or update it.
- Has up-to-date ARCore device profile data — if it is out of date, your app will download new data.

Set [ARSession.attemptUpdate](#) to `false` to disable these default checks. Your app should still handle these conditions:

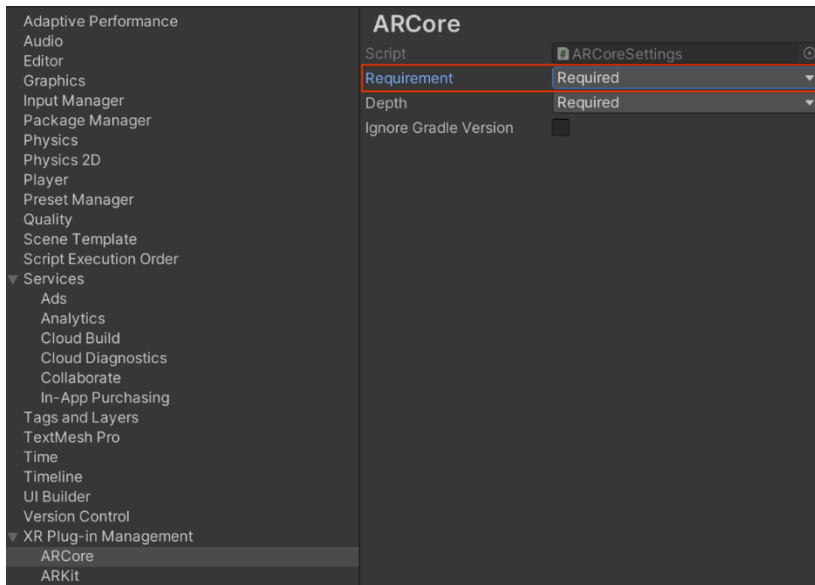
Condition	Action
Device does not support ARCore.	Hides AR-related functionality.
Device supports ARCore, but Google Play Services for AR and/or ARCore device profile data is not installed or out of date.	Prompts the user to install or update Google Play Services for AR and/or ARCore device profile data by calling ARSession.Install() .
Device supports ARCore and Google Play Services for AR and ARCore device profile data is installed and up to date.	Creates an ARCore <code>Session</code> and starts using ARCore.

You can also check the [ARSession.state](#) for installation and support states

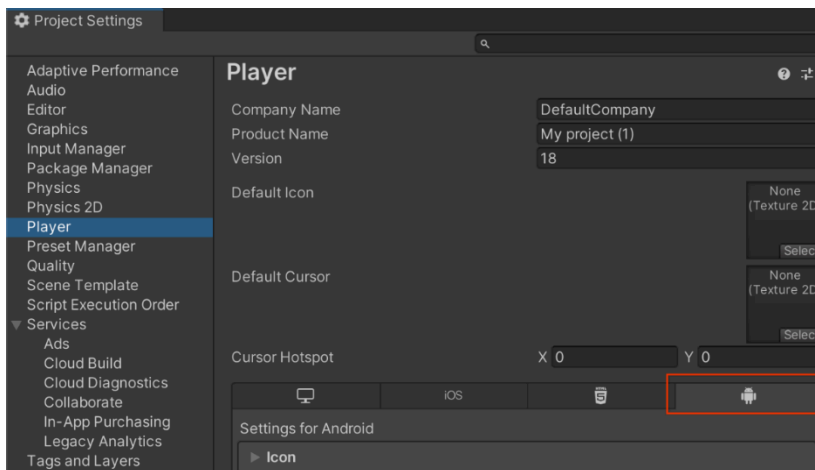
Android

1. Navigate to **Edit > Project Settings > XR Plug-in Management > ARCore. Requirement** is set to **Required** by default.

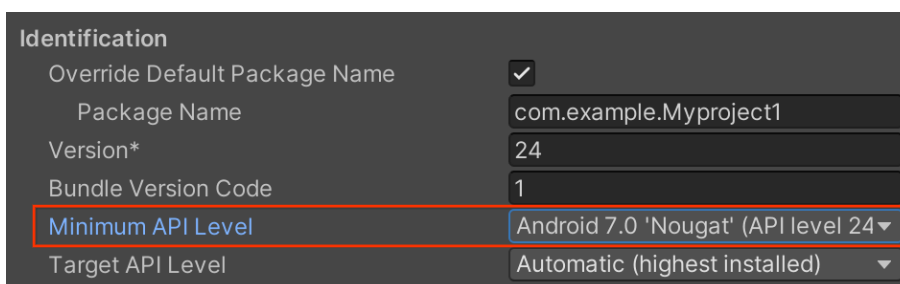
EXPERIMENT NO 7



2. Click **Edit > Project Settings > Player**, select the tab with the Android icon.



3. Under **Other Settings > Identification**, specify a **Minimum API Level** that is 24 or higher.

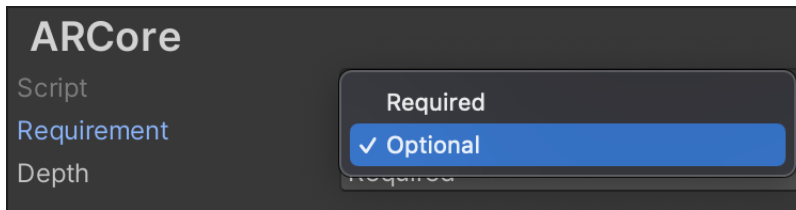


Make your app AR Optional

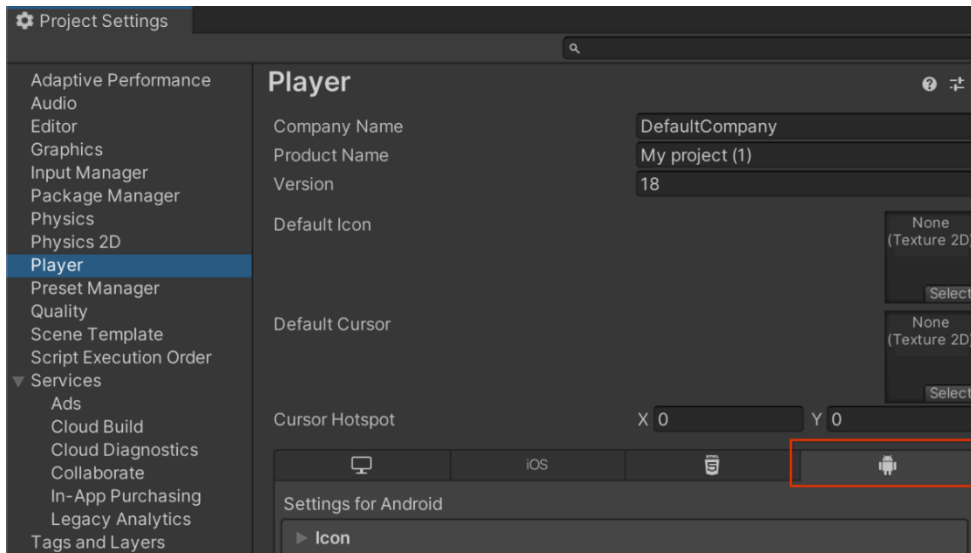
[AndroidiOS](#)

1. Navigate to **Edit > Project Settings > XR Plug-in Management > ARCore**.
2. From the **Requirement** drop-down menu, select **Optional** to set your app to AR optional.

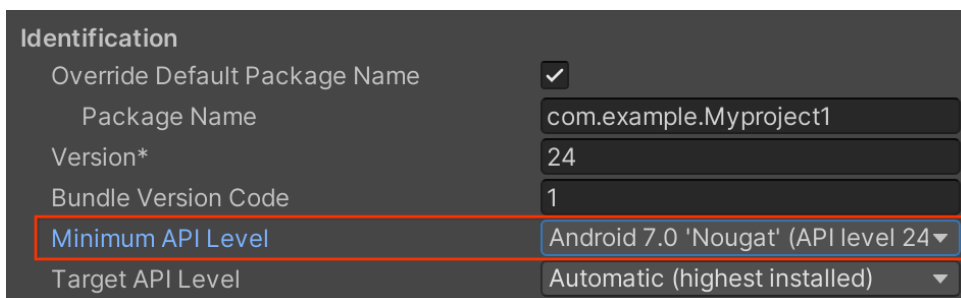
EXPERIMENT NO 7



3. Click **Edit > Project Settings > Player**, select the tab with the Android icon.



4. Under **Other Settings > Identification**, specify a **Minimum API Level** that is 24 or higher.



Perform runtime checks

Check if Google Play Services for AR is installed

Both AR Required and AR Optional apps must use the `ARSession.Install()` [coroutine](#) before creating an ARCore session to check whether a compatible version of Google Play Services for AR is (still) installed and to ensure that all required ARCore device profile data has been downloaded.

During runtime, AR Required apps will automatically perform runtime checks to ensure that the target device:

- Supports ARCore

EXPERIMENT NO 7

- Has installed Google Play Services for AR — if this is not installed (possibly because it was removed by the user) or the version is too old, your app will prompt the user to install or update it.
- Has up-to-date ARCore device profile data — if it is out of date, your app will download new data.

Set `ARSession.attemptUpdate` to `false` to disable these default checks. Your app should still handle these conditions:

Condition	Action
Device does not support ARCore.	Hides AR-related functionality.
Device supports ARCore, but Google Play Services for AR and/or ARCore device profile data is not installed or out of date.	Prompts the user to install or update Google Play Services for AR and/or ARCore device profile data by calling <code>ARSession.Install()</code> .
Device supports ARCore and Google Play Services for AR and ARCore device profile data is installed and up to date.	Creates an ARCore <code>Session</code> and starts using ARCore.

You can also check the `ARSession.state` for installation and support states

Get started with Geospatial Creator in Unity

bookmark_border

The ARCore Geospatial Creator for Unity allows you to preview Geospatial content in the Unity Editor by showing Google Maps data in a new 3D tiles format. This helps you visualize where content will be placed in the real world, while you are building your app.

The Geospatial Creator is available as part of the ARCore Extensions for AR Foundation. This Quickstart will show you how to use the Geospatial Creator in Unity to add geospatial AR content to the ARCore Geospatial Sample app.

If you have an existing app that you'd like to use instead of the Geospatial Sample, you'll first need to follow our [Geospatial Developer guide for Unity](#). After your app is configured for Geospatial API use, skip to the [Add Required Dependencies](#) step below and continue from there.

Prerequisites

EXPERIMENT NO 7

Follow ["Getting started with ARCore Extensions for AR Foundation"](#) to configure your Unity environment to create AR experiences.

While the ARCore Extensions supports earlier versions of Unity, the Geospatial Creator requires Unity 2021.3 or newer.

API Authorization

You'll need to set up API access to use the ARCore Geospatial API and the Google Map Tiles API.

Google Map Tiles API

The Map Tiles API provides a 3D, photorealistic representation of supported locations. We'll use these tiles in the Unity Editor to preview the area around your AR scene.

See the [Map Tiles API page](#) to enable the API and learn more about Map Tiles.

ARCore Geospatial API

The ARCore Geospatial API is used by your app at runtime to create and interact with Geospatial Anchors.

To enable the Geospatial APIs, set up the [ARCore API](#) in your application. You can use either the Keyless (recommended) or API Key authorization methods.

Extend the Geospatial Sample app

For this guide, we'll show you how to use the Geospatial Creator to add geospatial content to the ARCore Geospatial Sample app.

Configure the Geospatial Sample

First, familiarize yourself with the ARCore Geospatial Sample app. In a new Unity project, open the Unity Package Manager and select the ARCore Extensions package. In the Package Manager detail panel, expand **Samples** and import the Geospatial Sample into your Unity project.

Before running the app, you must enable ARCore and Geospatial functionality in your project. Use **Edit > Project Settings... > XR Plug-in Management**. Check the appropriate boxes to enable the ARCore plug-in provider for Android and/or the ARKit plug-in provider for iOS.

EXPERIMENT NO 7

Then, from **Edit > Project Settings... > XR Plug-in Management**, open the **ARCore Extensions** panel. Ensure "Geospatial" is checked (you don't need to enable the Geospatial Creator just yet), and enter your API credentials. See the "Enable the Geospatial Unity API" documentation for [Android](#) and [iOS](#) for more details.

There's a few other platform-specific Player settings that are required for any Unity app that uses ARCore. See the [Configure Player Settings](#) section of our docs for required changes.

At this point, there are no game objects in your scene other than what is required for running the Geospatial Sample. This is a good time to build and run the app. If configured properly, at runtime the app will display your device's camera view. As you move, the geospatial accuracy displayed at the top of the screen will begin to improve. By tapping on the screen, you'll create Geospatial Anchors that will persist at their original location while you move around.

Next, we'll show how to use the Geospatial Creator to create and visualize sophisticated AR content from within the Unity Editor. This content will appear in your app at specific latitude / longitude locations in the world.

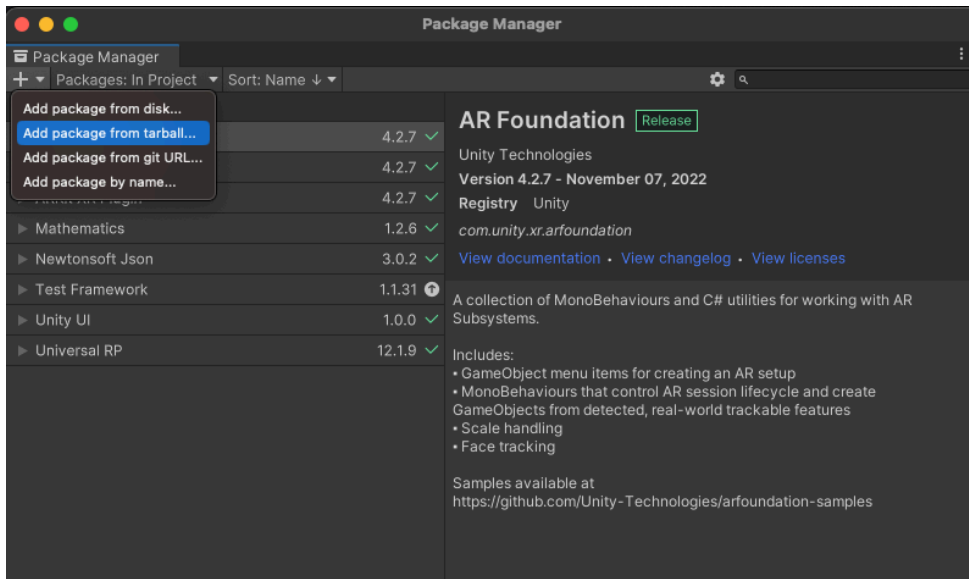
Add required dependencies

The Geospatial Creator makes use of a third party Unity package called [Cesium](#) to render Google Map Tiles in the Unity Editor's Scene window. Before enabling the Geospatial Creator features, let's add the Cesium package to your project.

We recommend installing the latest version of the Cesium for Unity package from Cesium's GitHub page:

1. Download the most recent version of the precompiled `.tgz` of Cesium for Unity from their [GitHub Releases page](#). The current recommended version of the Cesium for Unity package is 1.1.0.
2. Open the Unity Package Manager by clicking on Unity's "Window" menu, then selecting "Package Manager".
3. Click the "+" icon and choose "Add package from tarball".
4. Select the Cesium for Unity `.tgz` file downloaded in Step 1.
5. You may be prompted to install additional dependencies, such as Text Mesh Pro.

EXPERIMENT NO 7



Enable Geospatial Creator

You are now ready to enable the Geospatial Creator:

1. From the Build Settings dialog, make sure your selected platform is either "Android" or "iOS". Geospatial Creator features are not available for other platforms at this time.
2. Navigate to **Project Settings > XR Plug-in Management > ARCore Extensions**.
3. If you did not set your ARCore Android and/or iOS API key(s) in an earlier step, make sure to do that now.
4. Enable both "Geospatial" and "Geospatial Creator" toggles. After clicking to enable the Geospatial Creator toggle, a non-modal wizard will be displayed to confirm that your project is compatible with the Geospatial Creator.
5. Click "Finish" on the wizard.

After finishing the wizard, Unity will trigger a compilation of your scripts. Wait for the recompilation to complete before continuing. (If your selected platform is not "Android" or "iOS", switch to one of those now.)

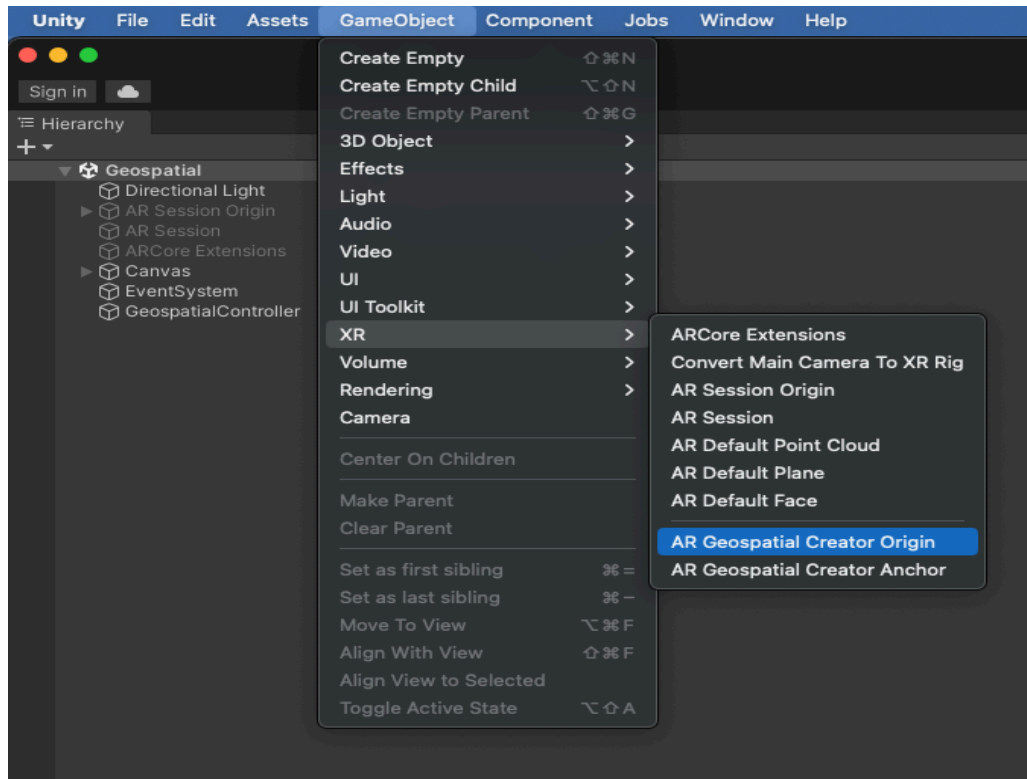
Add a Geospatial Creator Origin with Map Tiles

Your scene needs exactly one "Geospatial Creator Origin" object. This object is used as the reference point for converting real-world latitude, longitude, and altitude values to Unity game coordinates.

To add a Geospatial Creator Origin to your scene, select **Game Object > XR > Geospatial Creator Origin**. The newly-created object requires a georeference, which links a specific latitude, longitude, and altitude to the origin's position in game coordinates. From your Geospatial Creator Origin's **Inspector** pane, add a georeference by clicking the **Add Cesium Georeference Component** button.

EXPERIMENT NO 7

Note: If the **GameObject > XR** menu does not include **Geospatial Creator Origin**, ensure that you are building for Android or iOS, and you have enabled the Geospatial and Geospatial Creator toggles on the **Project Settings > XR Plug-in Management > ARCore Extensions** panel.



In addition to being the reference point for your AR objects, the Cesium Georeference adds a Cesium 3D Tileset object to the scene. This object will render Google Map Tiles in the Editor's Scene View, allowing you to visualize exactly where in the world your AR content will appear. There is a field on the Geospatial Creator Origin's inspector to add the Map Tiles API key that you [set up earlier](#). After you set this key, you should see the map tiles load in the Scene view.

Now change the latitude, longitude, and height fields to be located near where you'd like AR content to appear in your app.

Tip: You can find the latitude and longitude of most locations by finding that location on [Google Maps](#) and right-clicking. Selecting the latitude and longitude from the context menu will copy the values to your clipboard.

The Scene View in the Editor should now show the map tiles at your origin's specified latitude and longitude. At runtime, we assume your app will be using the AR Camera, so Geospatial Creator has already configured the Cesium 3D Map Tileset object with the "EditorOnly" tag. Because of this, the tiles will not be downloaded or rendered at runtime in your game. This can be changed from the Cesium3DTileset object's inspector. To learn more about Cesium's Georeference and 3D Tileset GameObjects, refer to the [Cesium for Unity documentation](#).

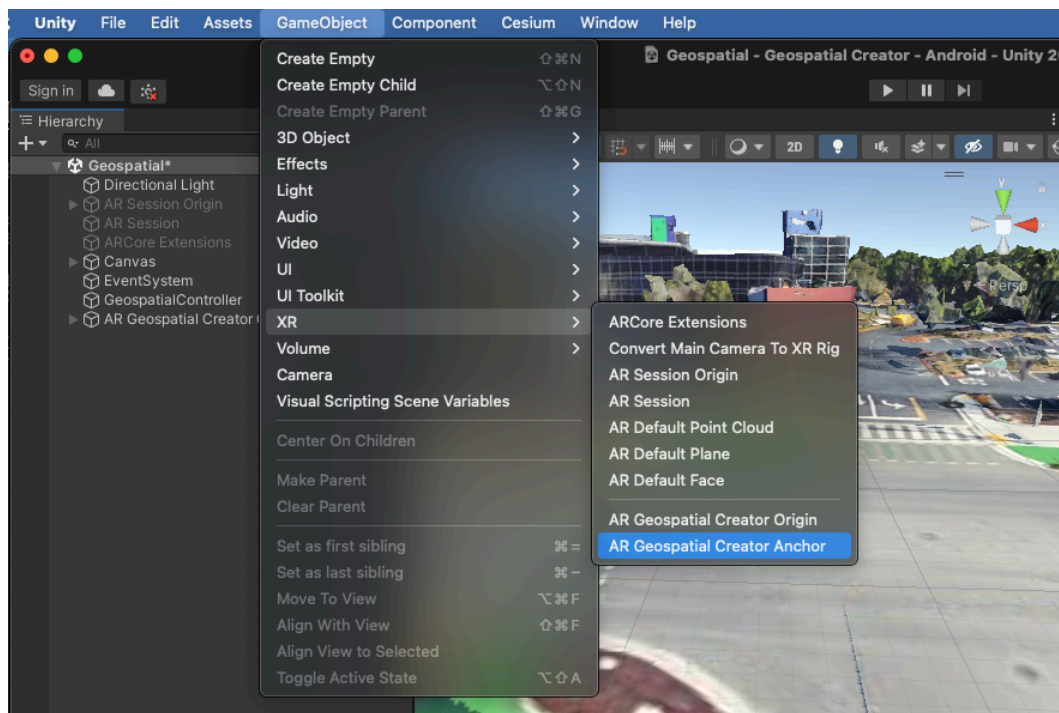
Tip: Your Scene View content, including the map tiles, may be obstructed by your application's UI elements. You can hide the UI layer in the Editor's Scene View, without

EXPERIMENT NO 7

disabling the UI at runtime, using the **Layers** drop down menu at the top-right corner of Unity Editor.

Add and place a Geospatial Creator Anchor

You are now ready to place AR content in your scene! Click **GameObject > XR > Geospatial Creator Anchor** to add an anchor to your scene hierarchy. From the Anchor's inspector, you can specify an exact latitude, longitude, and height. The object's transform will immediately update to the correct position in your scene, and you can see exactly where in the world the object will appear.



As you fine-tune the anchor's latitude and longitude, pick a place that is close by so you'll be able to validate it is in the correct place at runtime. When setting the altitude of a Geospatial Creator Anchor, you have three options:

- **Terrain** places the anchor at ground level at the latitude and longitude specified. The ground level is determined at runtime by the [Visual Positioning System](#). If you are new to ARCore's Geospatial APIs, "Terrain" is a good choice, because the anchor will be resolved directly on the ground with a high degree of vertical accuracy. The "Altitude Offset" field lets you position the anchor above or below ground level by a fixed amount, in meters. For example, specifying an offset of "1.0" causes the anchor to "float" one meter above the terrain.
- **Rooftop** places the anchor relative to the top of any structure at the given latitude and longitude. Like the terrain anchor, the "Altitude Offset" field lets you move the anchor up or down relative to roof position, in meters.
- **WGS84** places the anchor at a specific absolute altitude, in meters, based on the WGS84 coordinate system. Placing the object this way may require some trial and error, as a height of "0" in WGS84 does not necessarily indicate mean sea level and may not match a GPS's altitude. Also, while the latitude and longitude of the rendered 3D tiles is highly

EXPERIMENT NO 7

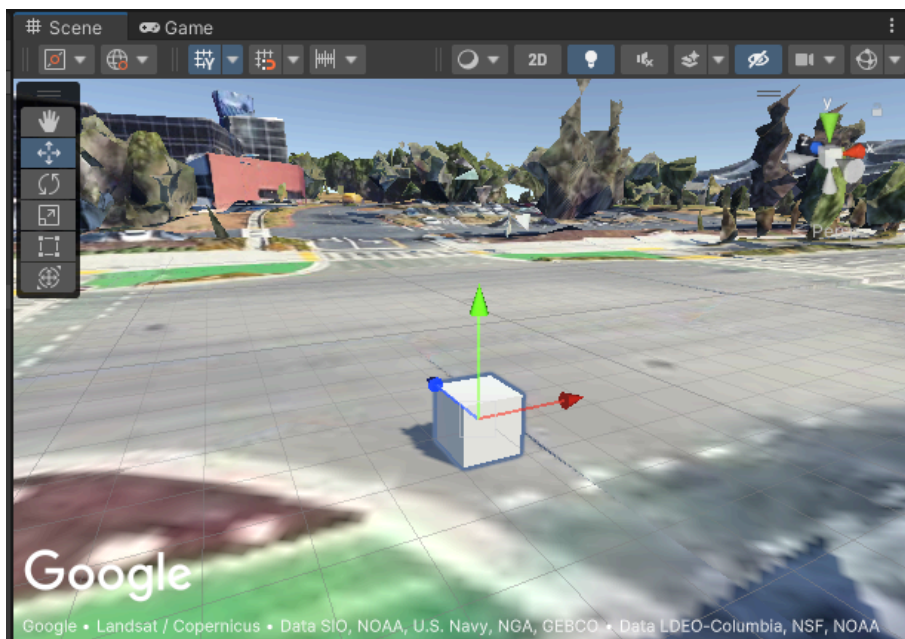
accurate, attempting to place the object at ground level using the 3D tiles as a reference could result in a height error of up to several meters.

To learn more about the different types of ARCore Geospatial Anchors, refer to the [Geospatial Developer Guide](#).

Both terrain and rooftop anchors have a **WGS84 Altitude** field. This value is *only* used in Editor mode, to set the height of the object in the Scene View. The terrain and rooftop heights are determined at runtime only, so the **WGS84 Altitude** field lets you visualize the anchor at a specific height while developing the app. The **WGS84 Altitude** field is ignored at runtime for terrain and rooftop anchors.

Add Geometry to the anchor

If you build and run the app now, you won't see the anchor in your AR enabled app. That's because there's no geometry attached to it. Let's add a simple 3D primitive to the anchor: In the scene hierarchy, right click on your AR Geospatial Creator Anchor object. From the context menu, choose **3D Object > Cube**. You should see a gray cube in your scene view as a child of your anchor. If you decide to move the object's position or rotation in the Editor, we recommend moving the root Anchor object instead of the child Cube primitive. This will help reduce error in the translation from latitude / longitude to game coordinates.



Of course, you can add more complicated geometry and arbitrary components and children to the anchor as well.

EXPERIMENT NO 7

Build and run your app

It is time to see your anchor in the real world! Try building and running your app, then move to the location where you expect to see the cube. Is it there? Congratulations, you're ready to build more complex AR experiences with the Geospatial Creator!

If you have any issues, see the [Troubleshooting Guide](#).

Now is a great time to add additional AR Geospatial Creator Anchors with different geometries. Try experimenting with multiple rooftop and terrain anchors and their offsets. Add scripts to animate or enable interactions with the objects.

Understanding Geospatial Creator Anchors and the `ARGeospatialAnchor` class

When constructing complex and interactive AR experiences, it is helpful to understand the relationship between Geospatial Creator Anchors and ARCore's `ARGeospatialAnchor` C# class:

An [ARGeospatialAnchor](#) can only be created at runtime by invoking one of the functions on [ARAnchorManager](#), and they can only be created once the AR session has stabilized. A Geospatial Creator Anchor can be thought of as a placeholder that exists in your scene, waiting for the AR session to be ready to resolve `ARGeospatialAnchor` objects. Once the session is ready, the placeholder will automatically call the `ARAnchorManager` to create and resolve the `ARGeospatialAnchor`.

Immediately after the runtime anchor is resolved, the Geospatial Creator Anchor in your scene hierarchy will attach itself as a child to the new `GameObject`. This means that any components or children of the Geospatial Creator Anchor will become subcomponents or grandchildren of the new `ARGeospatialAnchor` `GameObject`. For example, the `transform` of a geometry primitive attached to the Geospatial Creator Anchor will inherit from the `transform` of the `ARGeospatialAnchor`, and will therefore be placed at the expected location in the runtime AR session.

If you need to identify specific `ARGeospatialAnchor` `GameObjects` at runtime, you can search the scene for objects whose `name` property matches the anchor created using the Geospatial Creator. If the AR session is initialized, the `parent` of that object will be the corresponding `ARGeospatialAnchor` object.

Troubleshooting

EXPERIMENT NO 7

Why is my anchor moving?

You may notice, especially when the app first launches, that the anchor may appear to slowly "drift", or even jump several meters abruptly. This occurs when the VPS session is still determining and refining the precise location of the camera. As the horizontal and vertical accuracy improves over multiple frames, the anchor is redrawn at a more accurate location, giving the illusion of movement. You may wish to render the object differently (or not at all) until the location accuracy reaches a certain threshold. To do so, you can query the [`ARCoreEarthManager.CameraGeospatialPose`](#) property to determine the accuracy for the current frame update.

Why does my anchor appear at the wrong altitude?

When using the "Manual" altitude type for Geospatial Creator Anchors, the anchor's altitude will be set (in meters) according to [`WGS84`](#). At a given location, there may be a discrepancy of up to several meters between the apparent ground level shown in the map tiles and the actual ground level as reported by [`Google's Visual Positioning System \(VPS\)`](#). If a Geospatial Anchor appears to be located incorrectly at runtime, we recommend validating the VPS height in person using the debugging overlay in the sample app. You can then change the altitude in the Unity Editor to match the known VPS height at that location.

Map tiles are not rendering properly in the Editor's Scene View

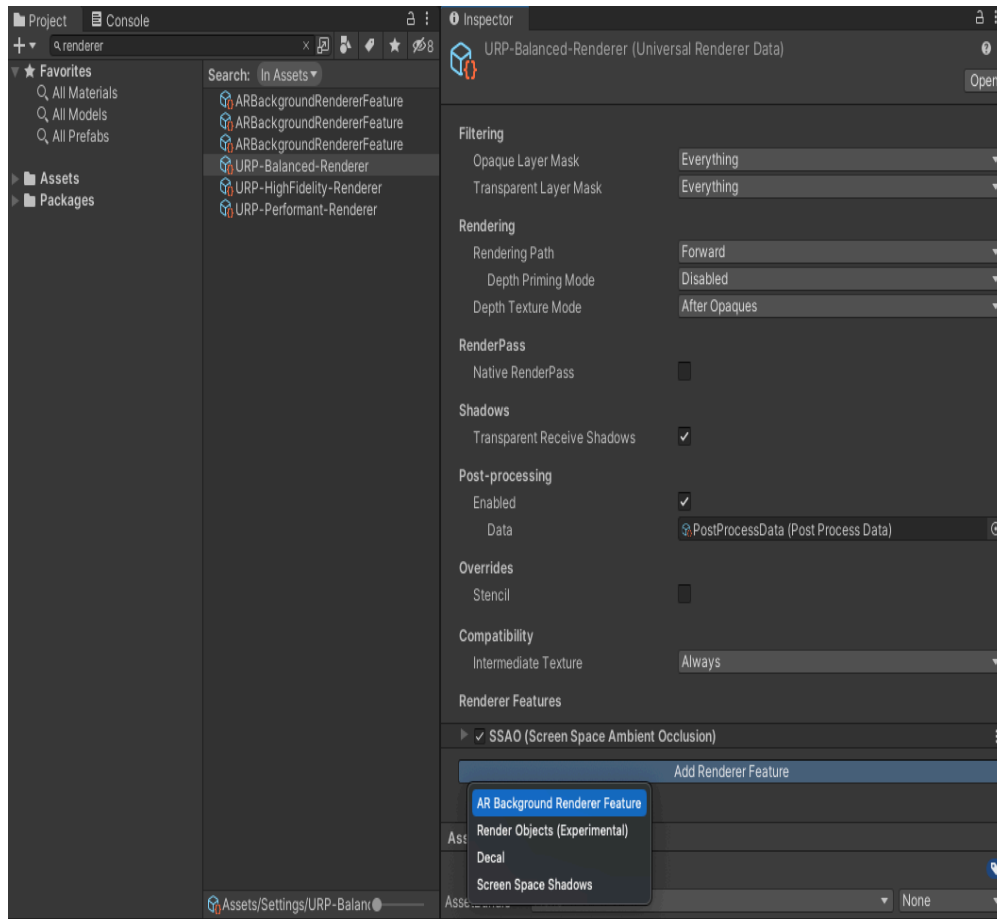
If the `Cesium3DTileset` appears to be loading tiles, but they are rendered in solid magenta in your scene view, add Unity's Shader Graph package to your project. This package is required by Cesium, though projects using Unity's built-in renderer do not include it by default.

To add Shader Graph, open the Unity Package Manager (**Window > Package Manager**). Make sure you are browsing packages available on the Unity Package Registry by selecting "Packages: Unity Registry" from the dropdown menu at the top of the Package Manager window. Search for the `com.unity.shadergraph` package and install the latest version. Once the package loads, the 3D tiles should reload and render properly in the Scene View.

The app displays a black screen at runtime instead of the camera view

Unity's Universal Rendering Pipeline (URP) is not configured for the AR Camera by default. If you are seeing a black screen at runtime instead of the camera output, you'll need to add the `AR Background Renderer Feature` to each URP Renderer in your scene.

EXPERIMENT NO 7



To add the feature, search for "renderer" in your project. For each URP Renderer object, click the "Add Renderer Feature" button in the Inspector and add the AR Background Renderer.

Why do Anchors placed far from the origin seem to be placed incorrectly?

The conversion from WGS84 coordinates to Unity game coordinates is subject to error due to floating point arithmetic precision, which can be minimized by ensuring your origin is nearby to your AR anchors. Distances over 20km may lead to inaccuracies. We recommend using multiple scenes, with a unique anchor in each scene, if your AR experience spans long distances.

Was this helpful?

EXPERIMENT NO 7

CONCLUSION :

- ARCore enables motion tracking, surface detection, and light estimation for AR experiences.
- Unity with AR Foundation and ARCore XR Plugin is required for setup.
- AR Session and AR Camera are essential to run AR applications.
- Apps can be built as AR Required or AR Optional.
- Geospatial Creator helps place AR objects in real-world locations.