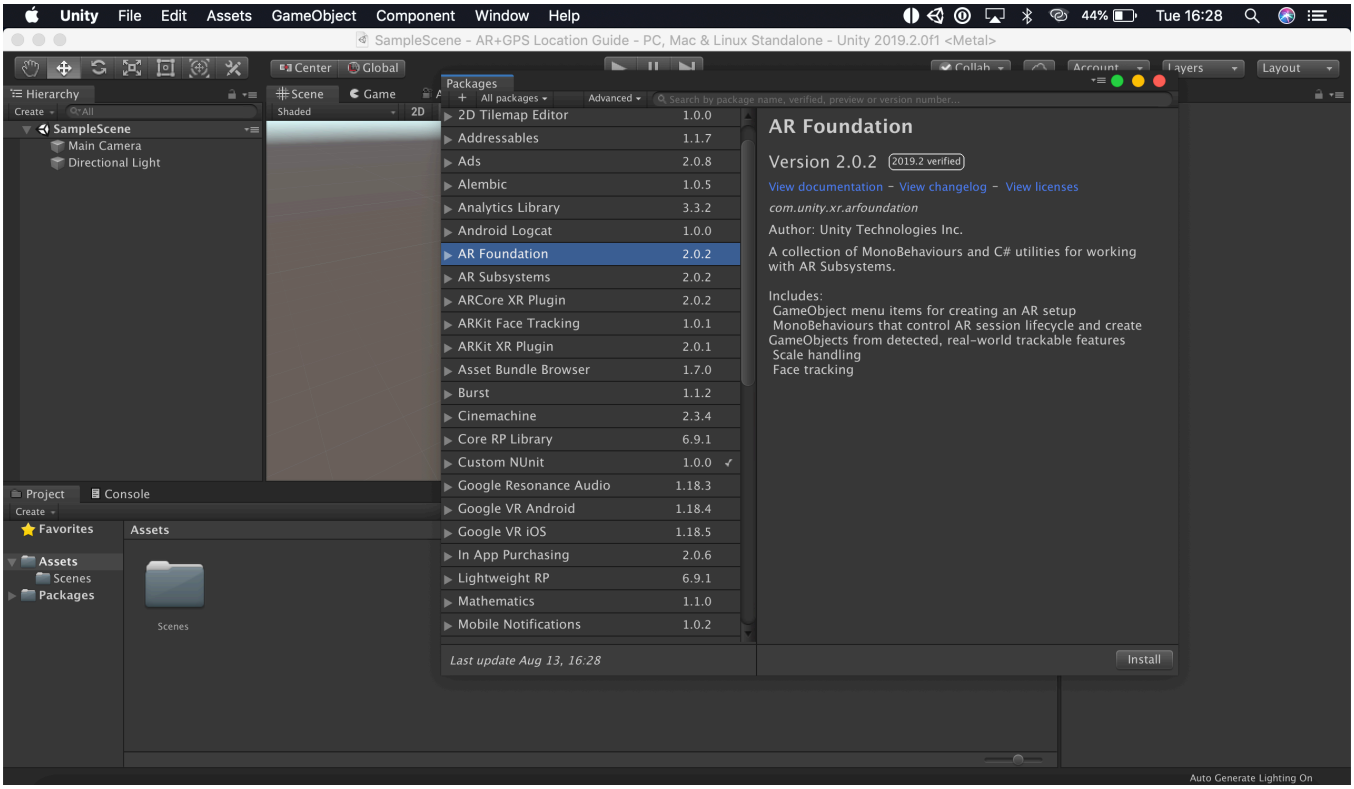
# # Quickstart

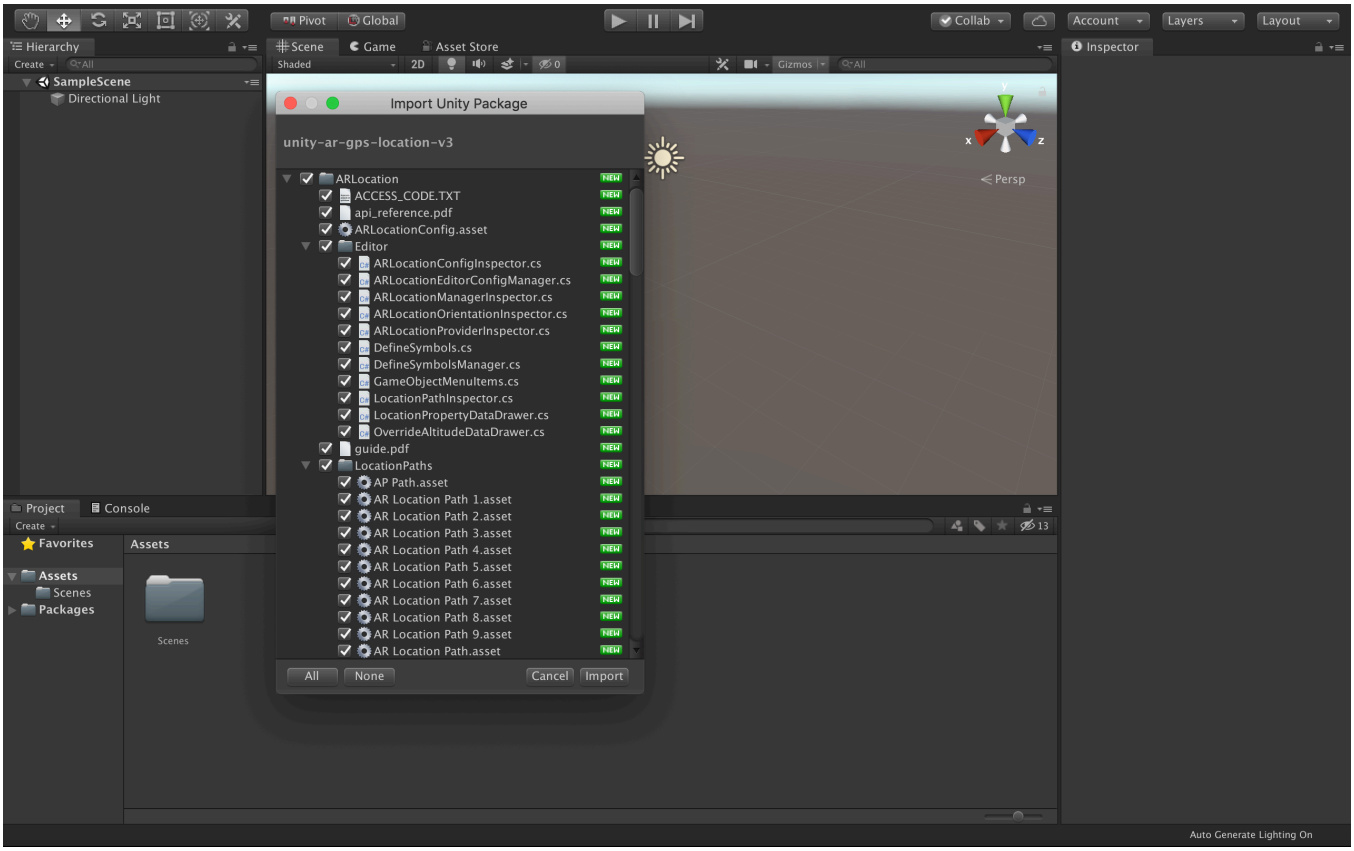
## # AR Foundation

## # Guide

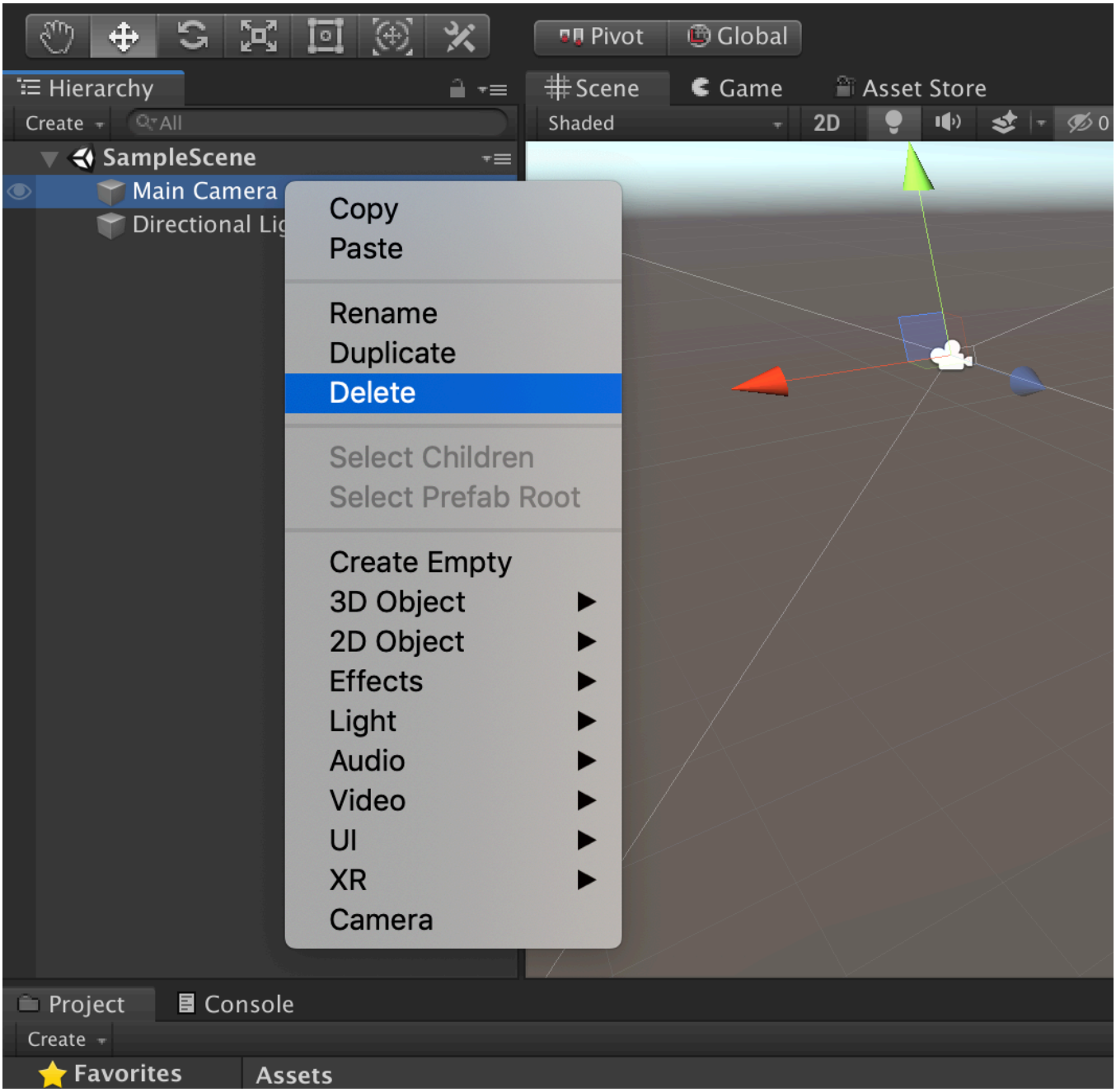
Either starting from an empty project or an existing one, open the `Package Manager` window by clicking on `Window -> Package Manager` and install the `AR Foundation` package, as well as the `AR Core XR Plugin` and/or the `AR Kit XR Plugin`, depending on which platforms you are targeting.



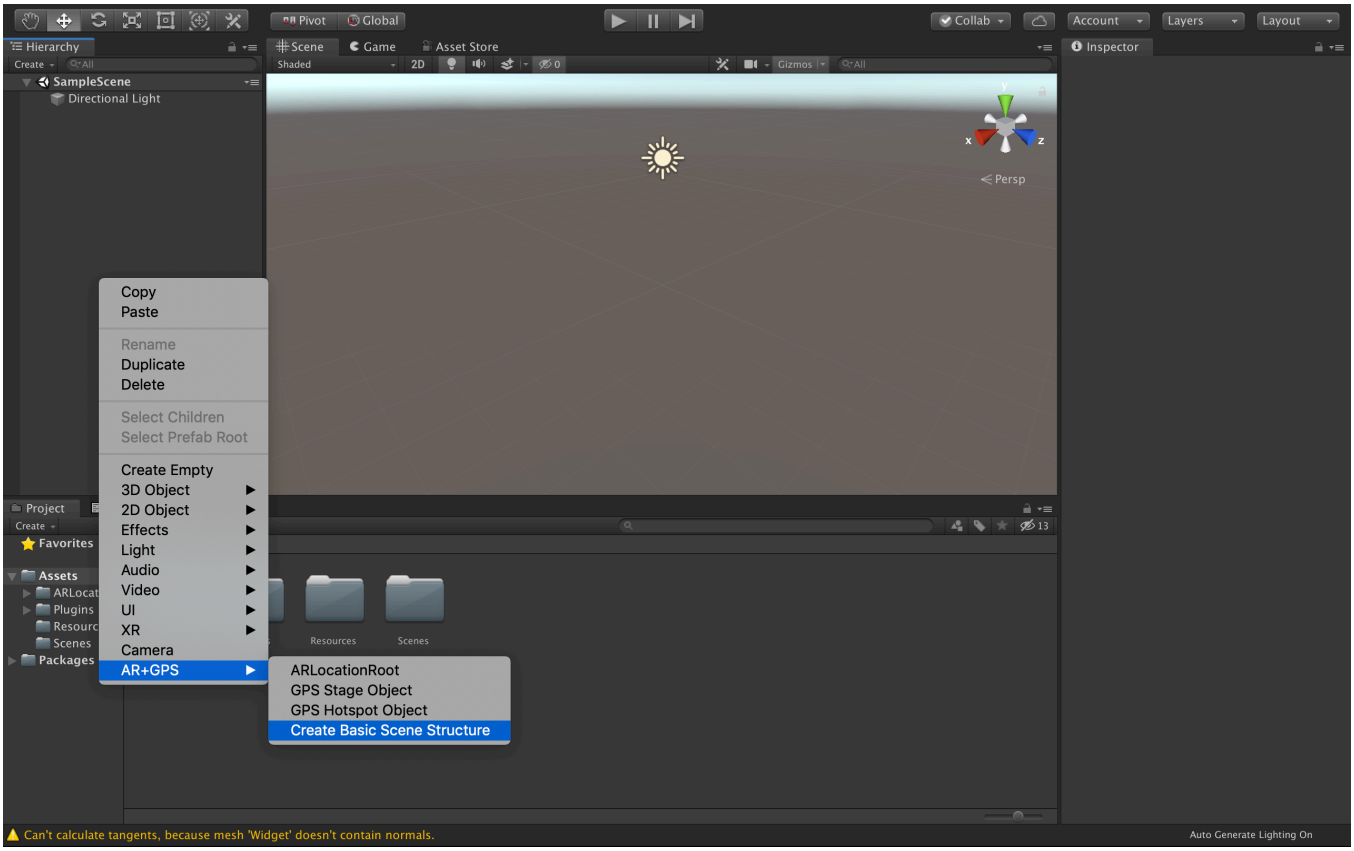

Next, import the `Unity AR+GPS Location` package, either from the asset store, or from a `.unitypackage` you have downloaded from our github repository.



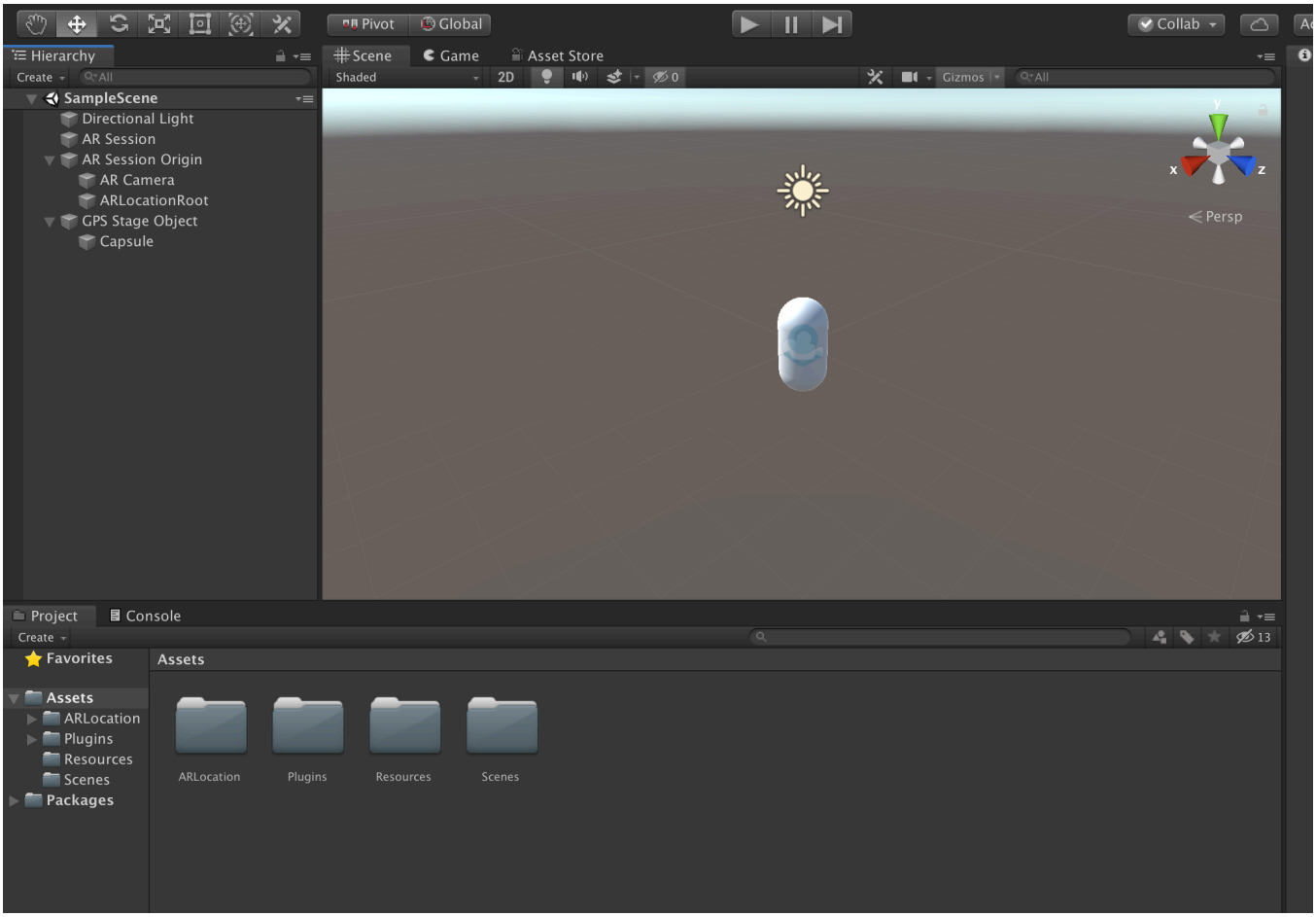

Then, remove the default 'Main Camera' from the scene.



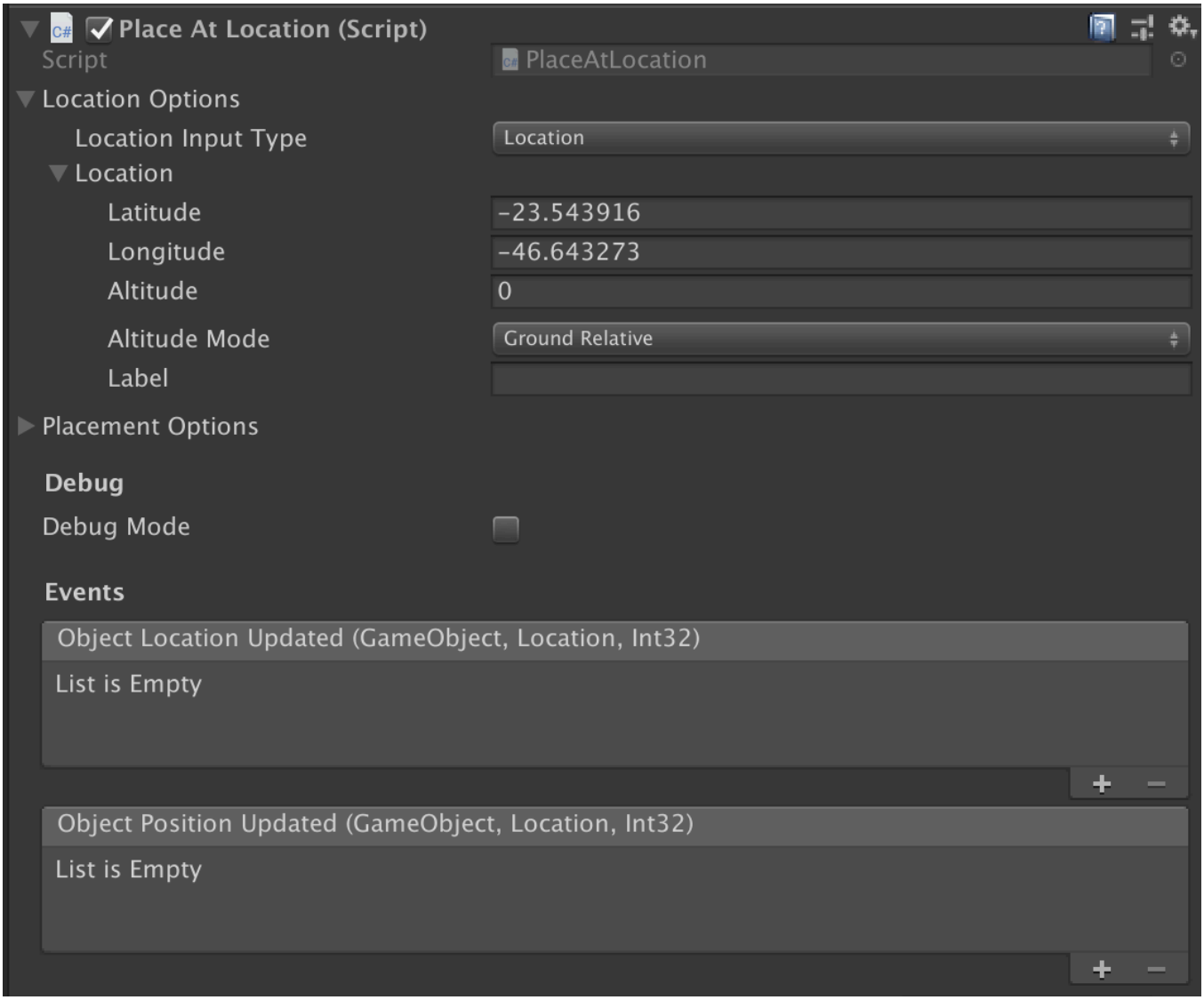

Right-click on the `Hierarchy` and go to `AR+GPS -> Create Basic Scene Structure`.



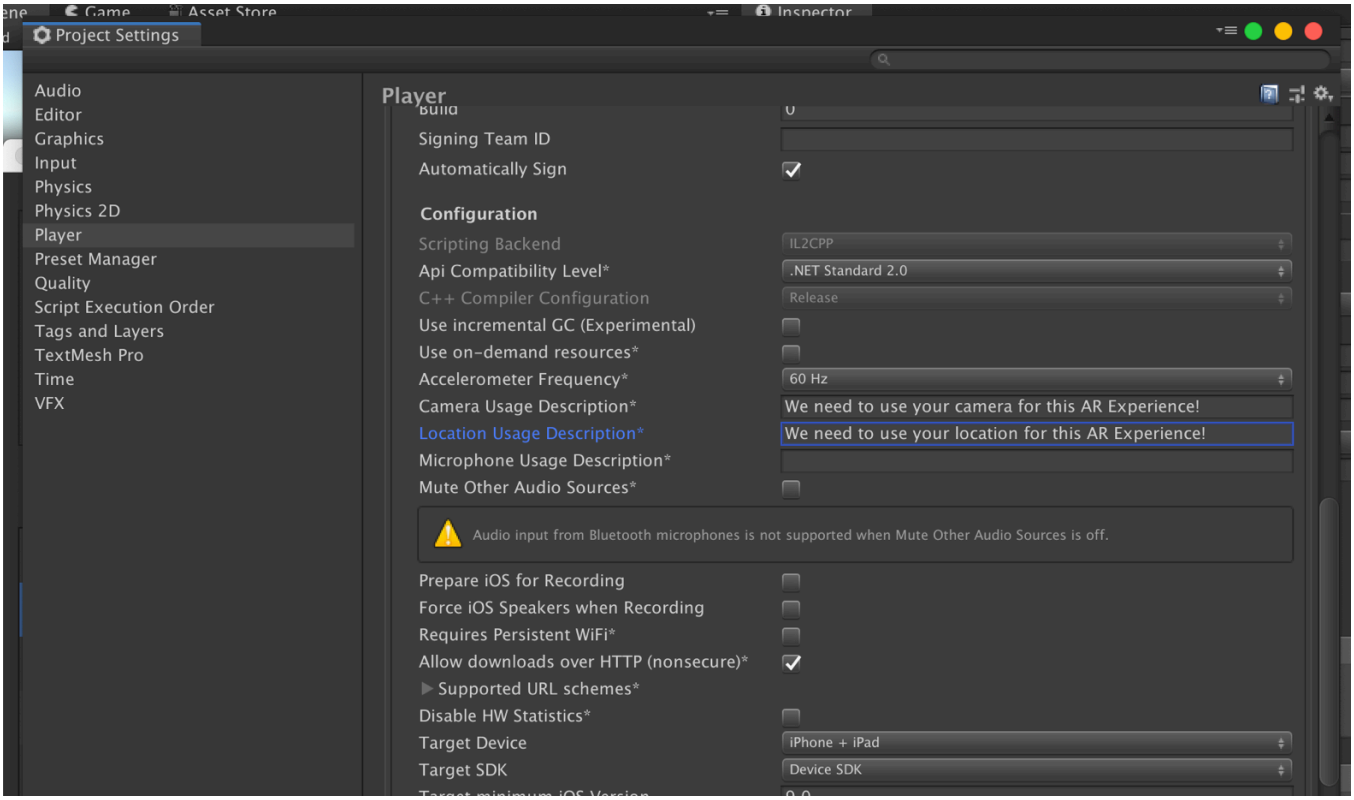

This will automatically populate the scene with all the Game Objects and Components for a basic AR+GPS experience!




Now just click on the `GPS Stage Object` and on the `Inspector` window, go into `Location Options -> Location`, and put the geographical coordinates of the location you want the object to appear at.



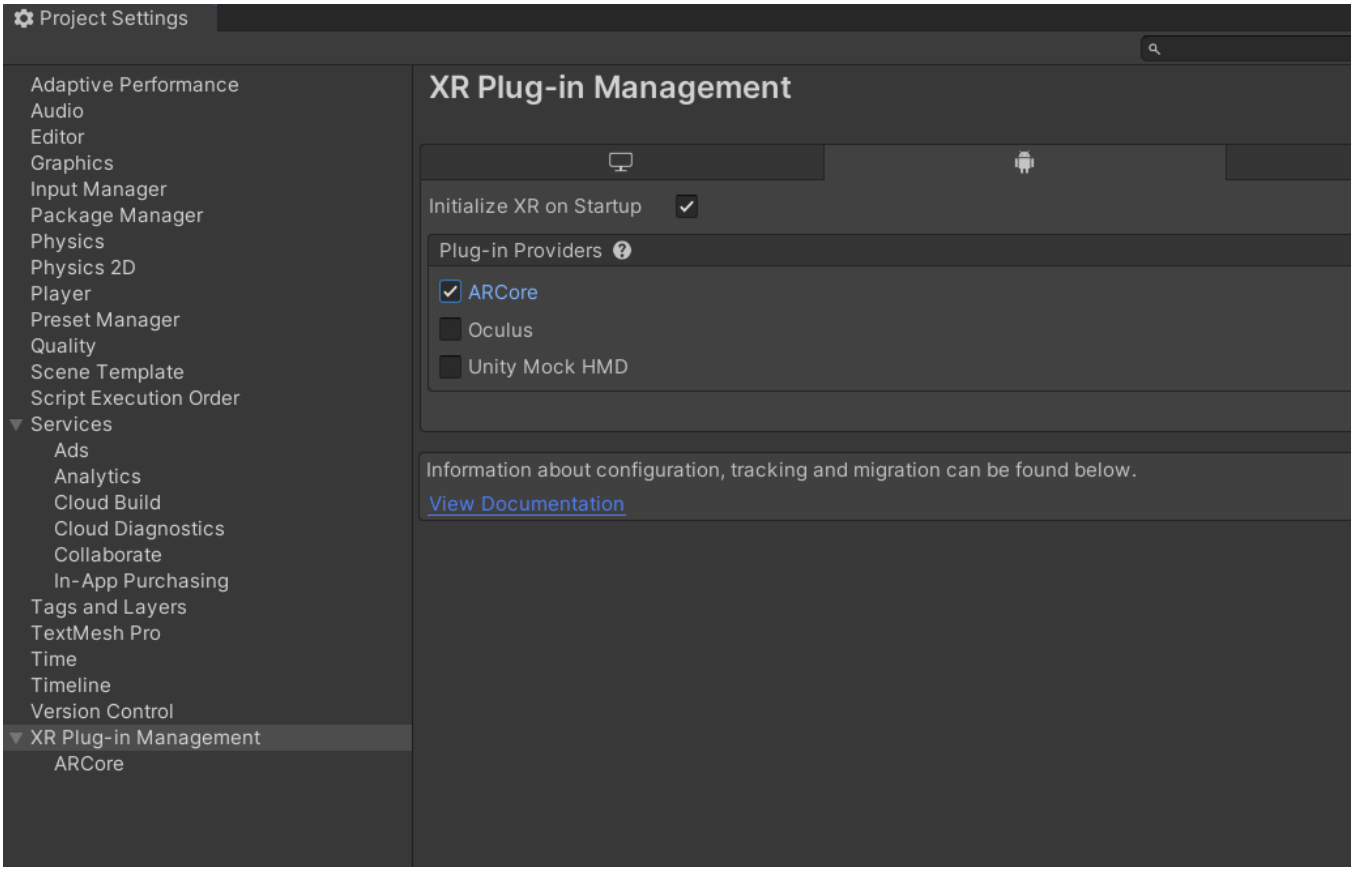

That's about it! Now you just need to build the project. On iOS just remember to set the `Camera Usage Description` and `Location Usage Description` strings on the Player Settings, and set the `Achitecture` to `ARM64`.




Now build and deploy the project onto your device, and you should the the object at the location you placed it at!

## Note

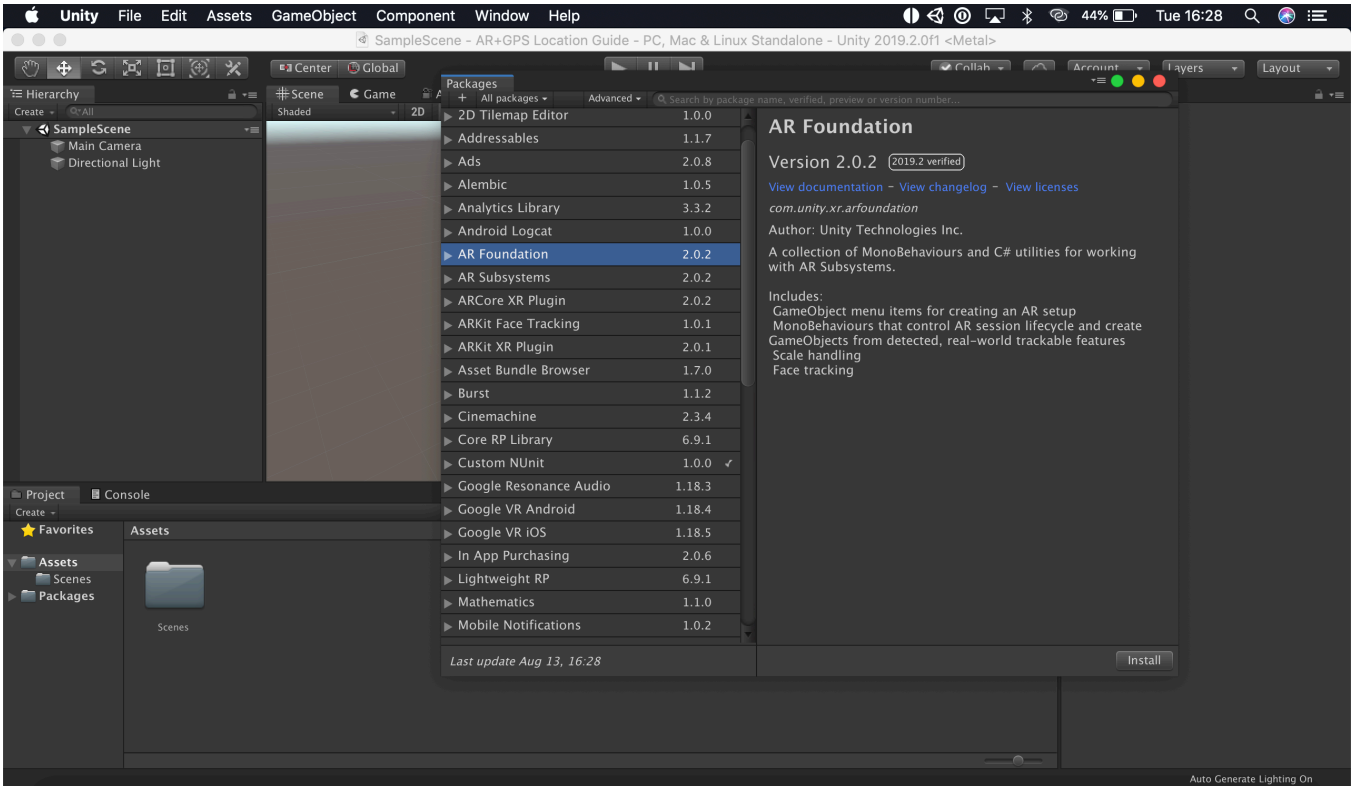
If you are using AR Foundation 4.0 or newer, you will also need to configure the Plugin Provider. To do that go to the `Project Settings` window, and then to `XR Plug-in Management`.



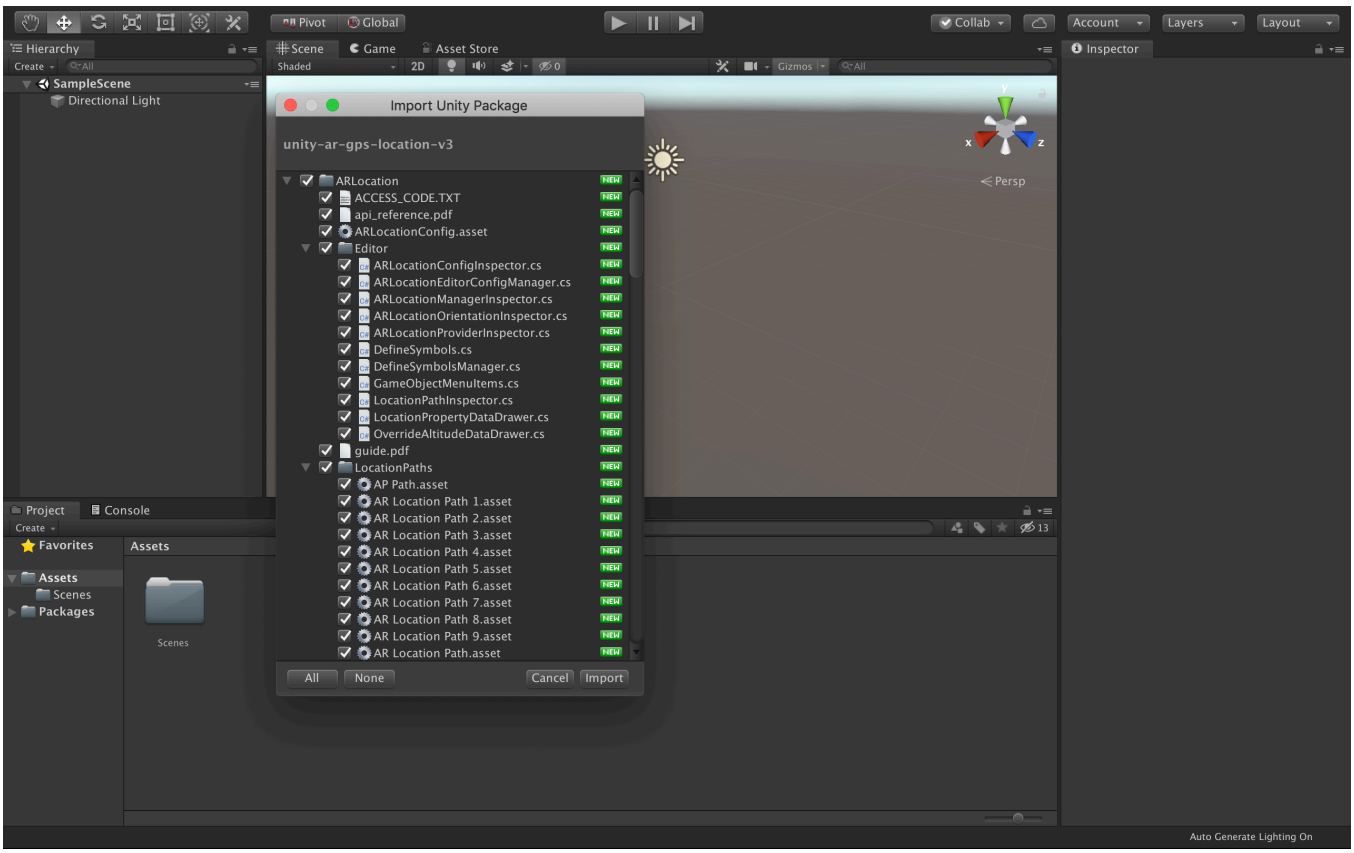

For more information on how to setup AR Foundation, please refer to the AR Foundation manual.

## # Vuforia

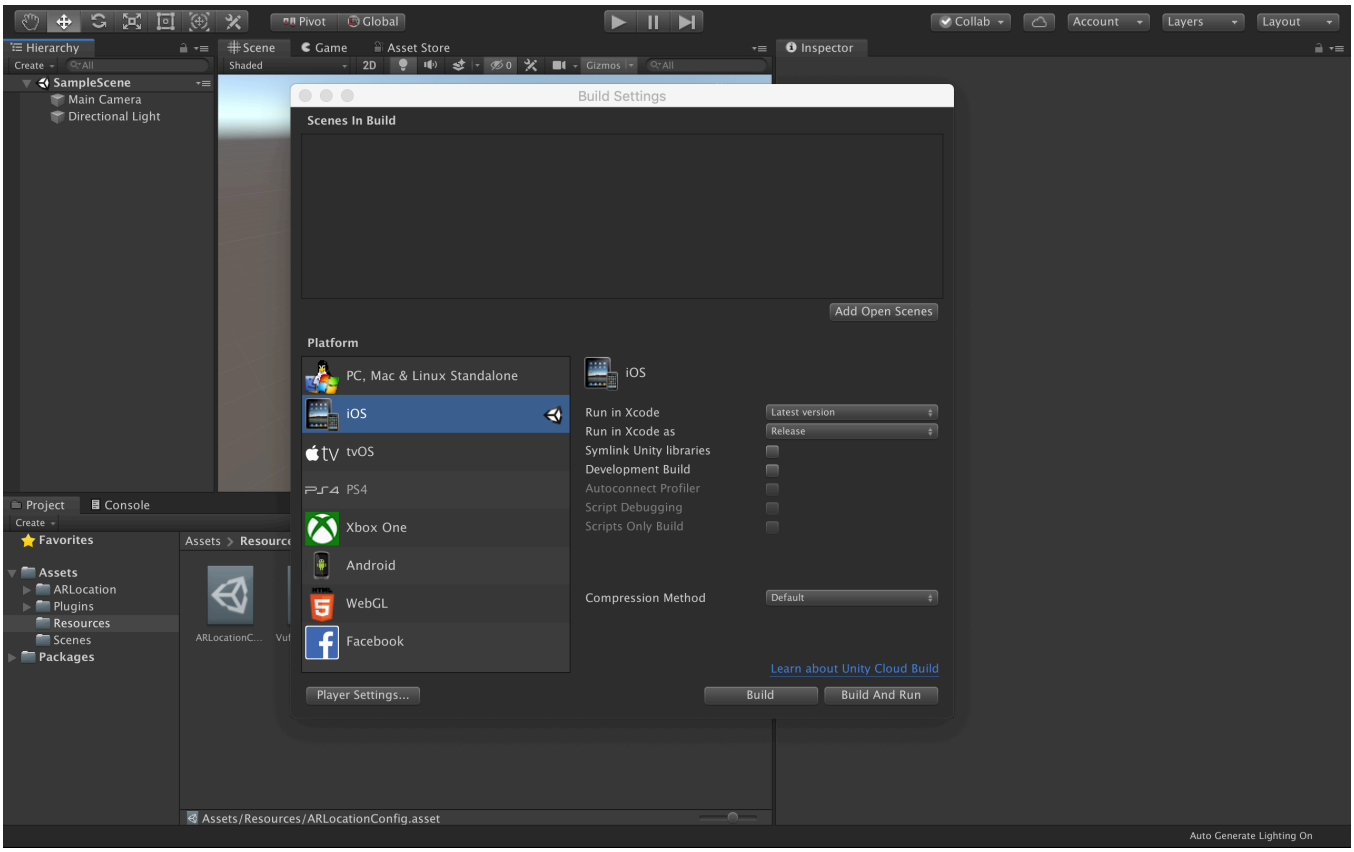
For Vuforia, starting with an empty project, open the `Package Manager` window by clicking on `Window -> Package Manager` and install the `AR Foundation` package. This is necessary to avoid scripting errors in the project, even though `AR Foundation` is not actually used here.



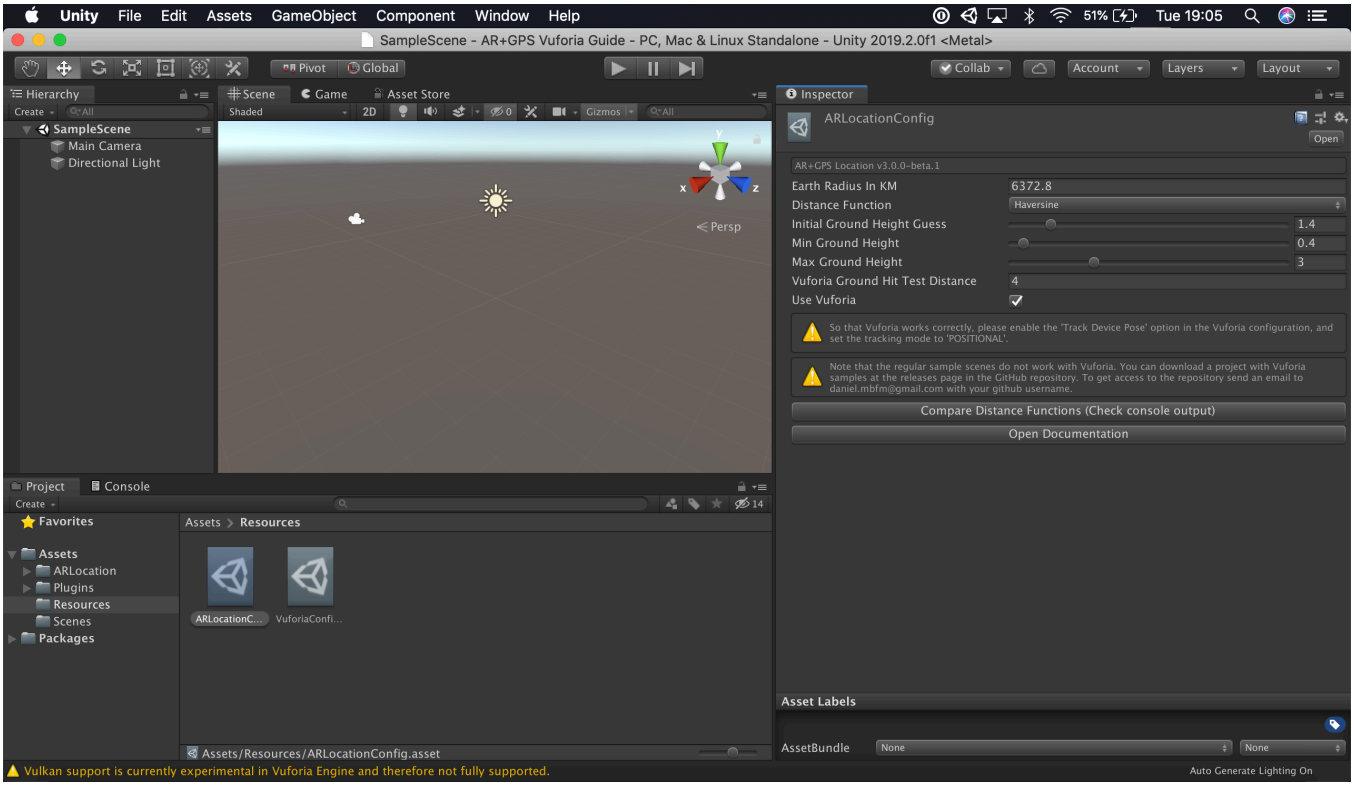

Next, import the `Unity AR+GPS Location` package, either from the asset store, or from a `.unitypackage` you have downloaded from our github repository.



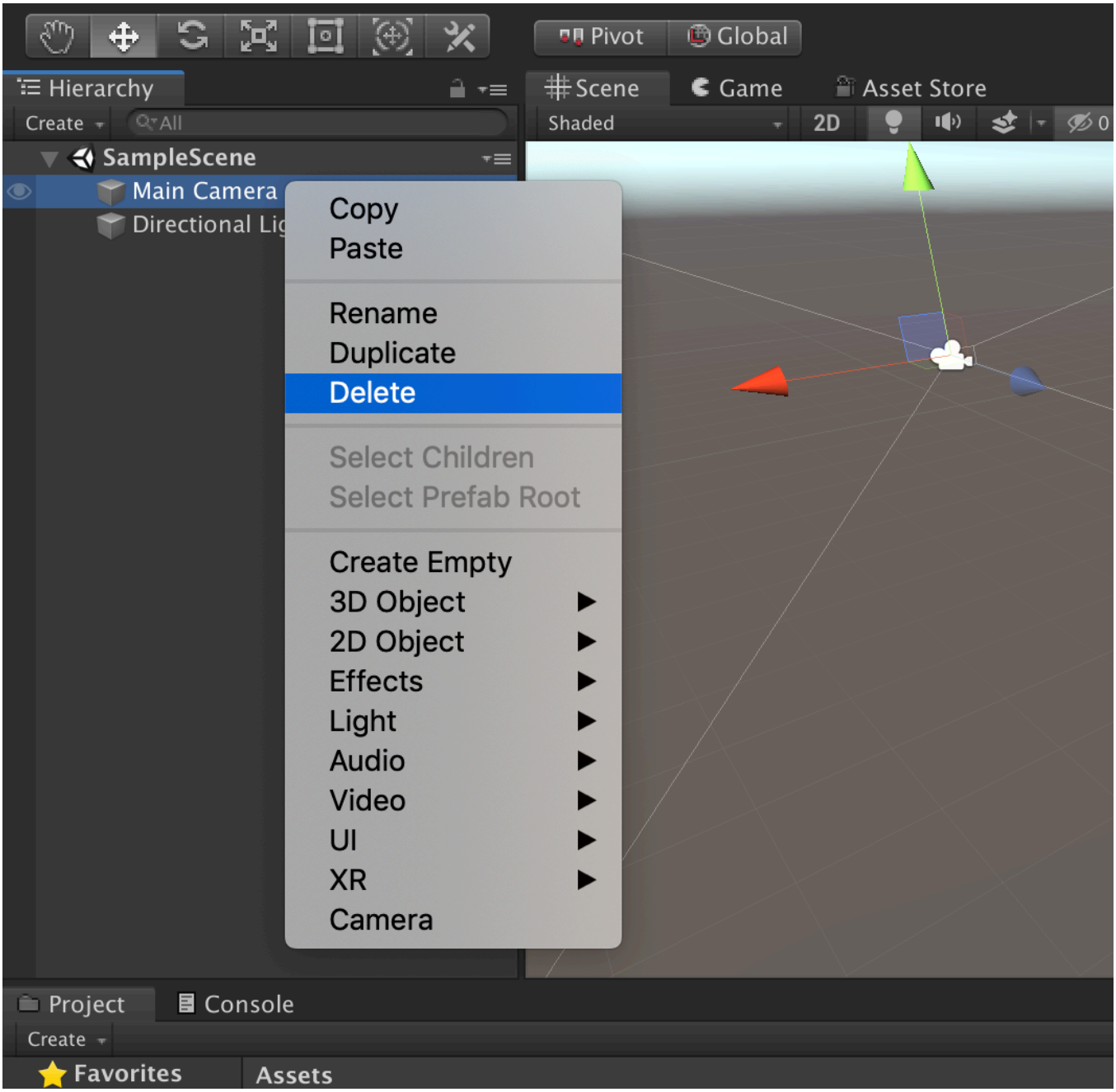
Make sure the current platform is set to either `Android` or `iOS` in the `Build Settings` window.



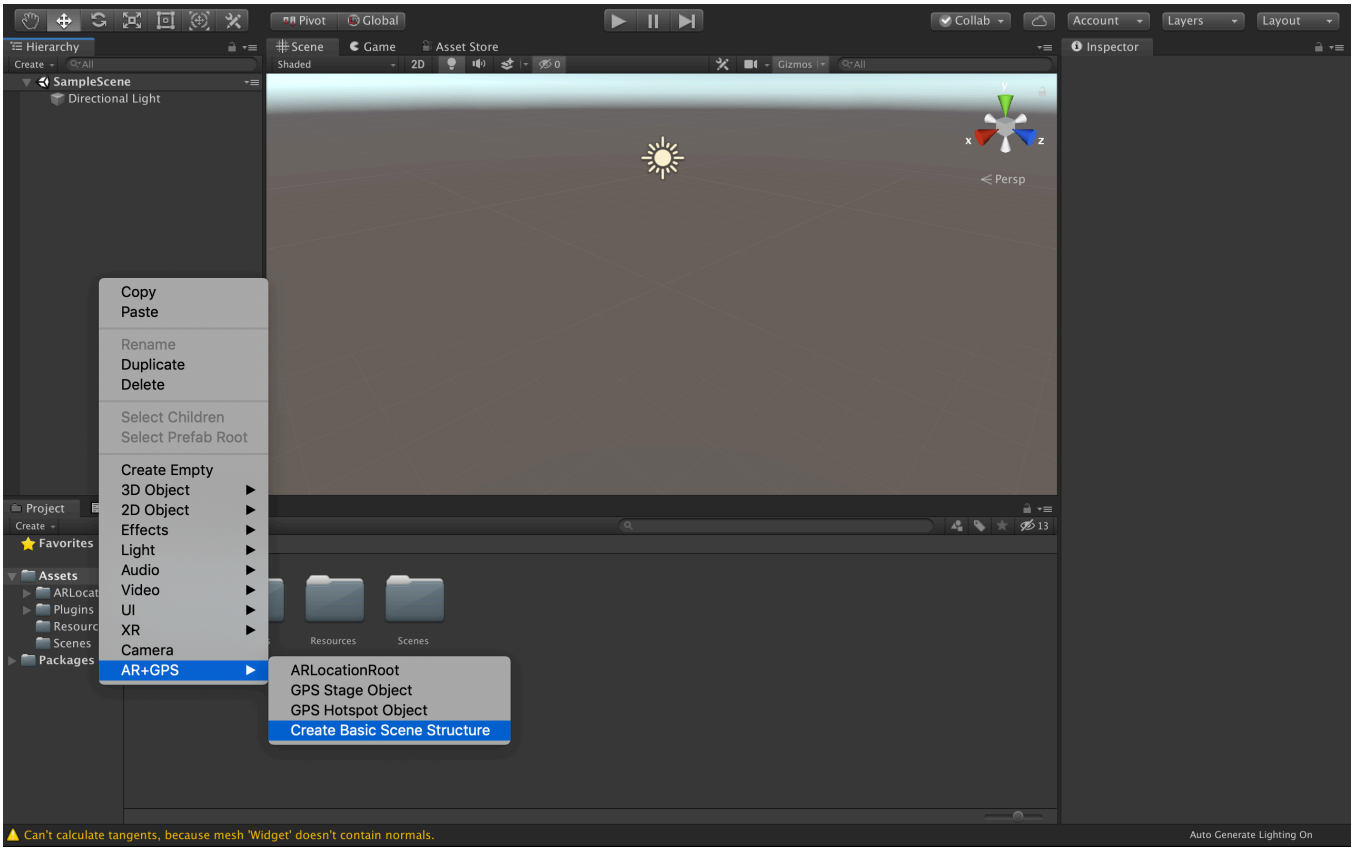

After that, navigate to the `Resources/ARLocationConfig` asset in the `Project` window, check the `Use Vuforia` option in the inspector, and wait for `Vuforia` to be imported into the project.



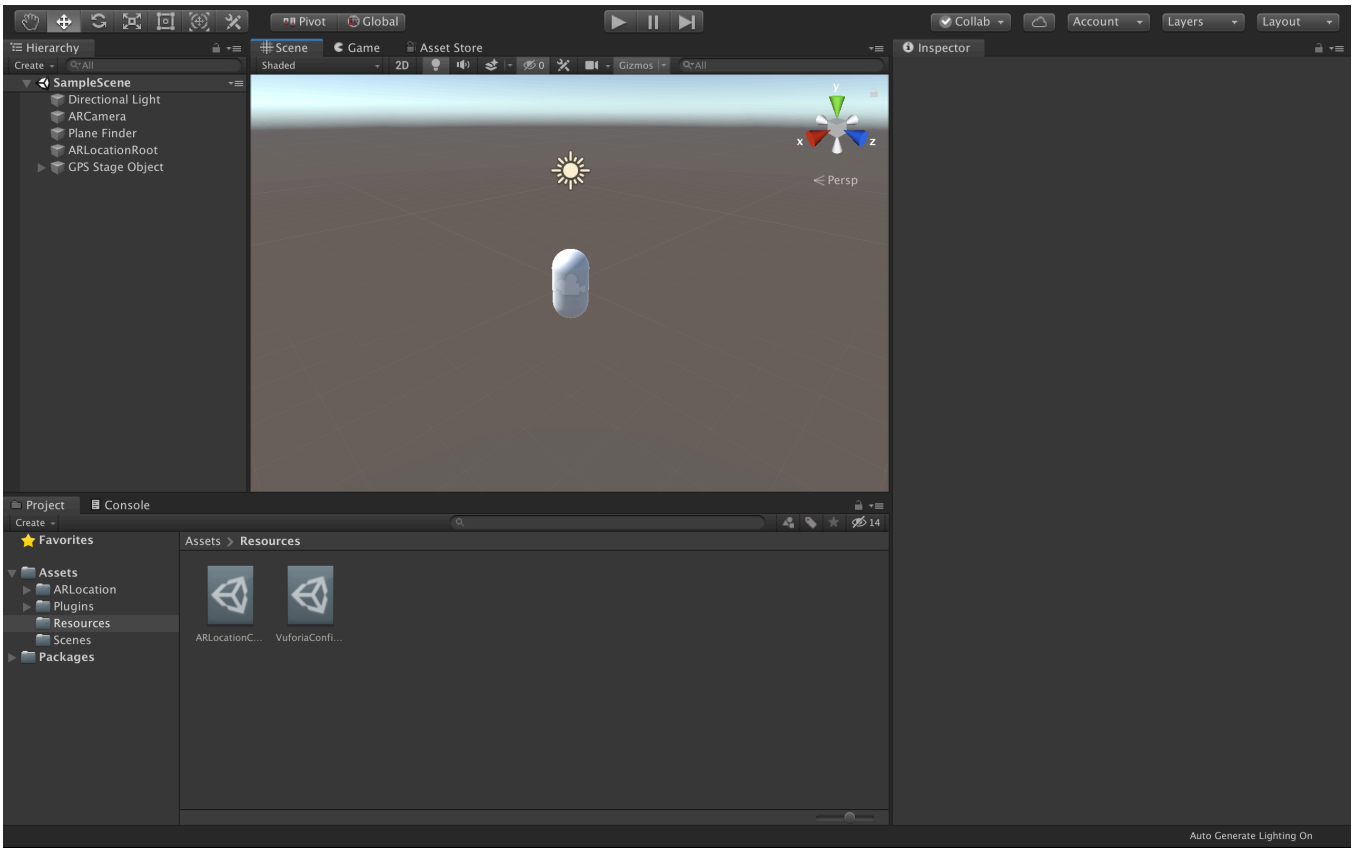

Then, remove the default 'Main Camera' from the scene.



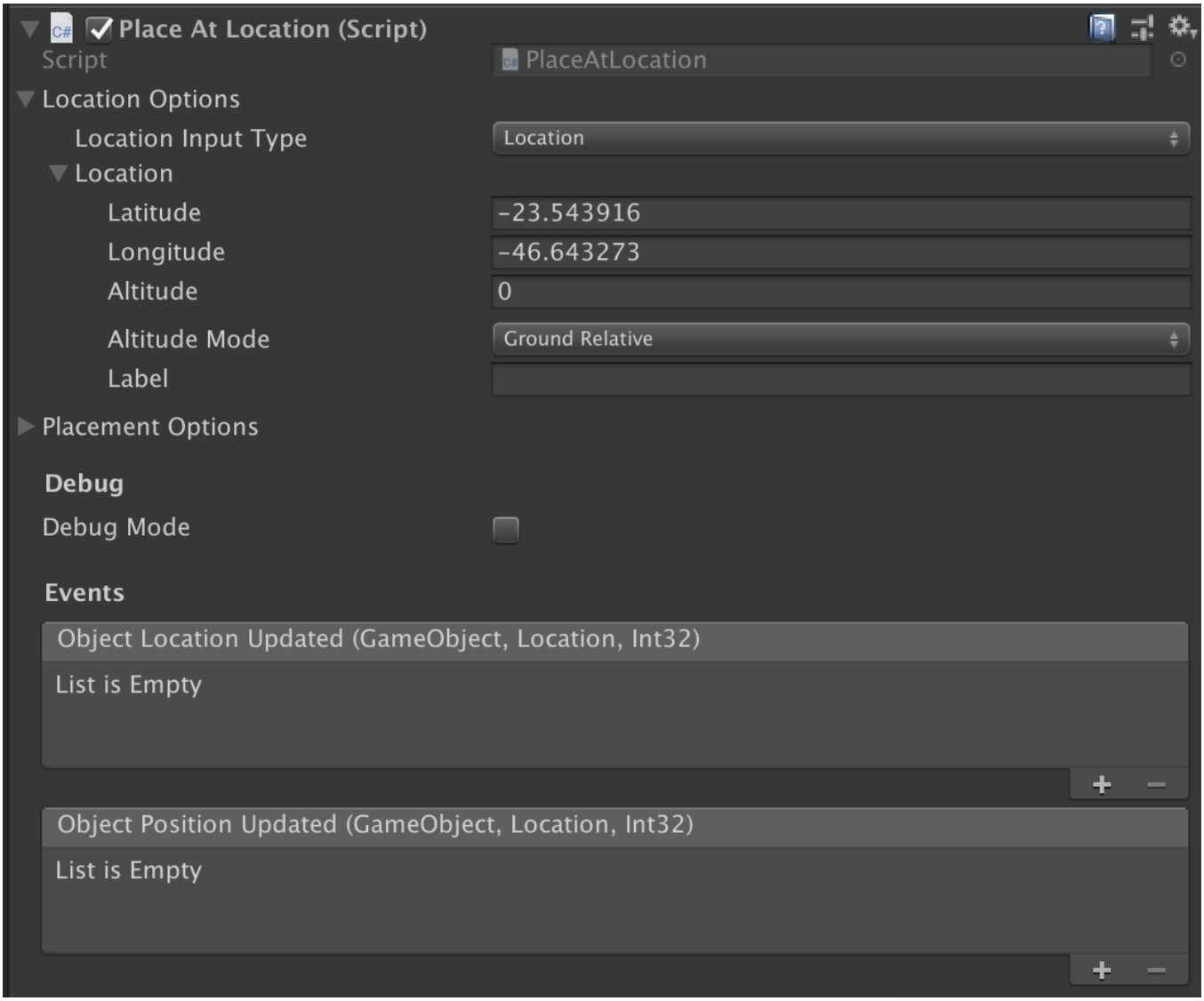

Right-click on the `Hierarchy` and go to `AR+GPS -> Create Basic Scene Structure`.



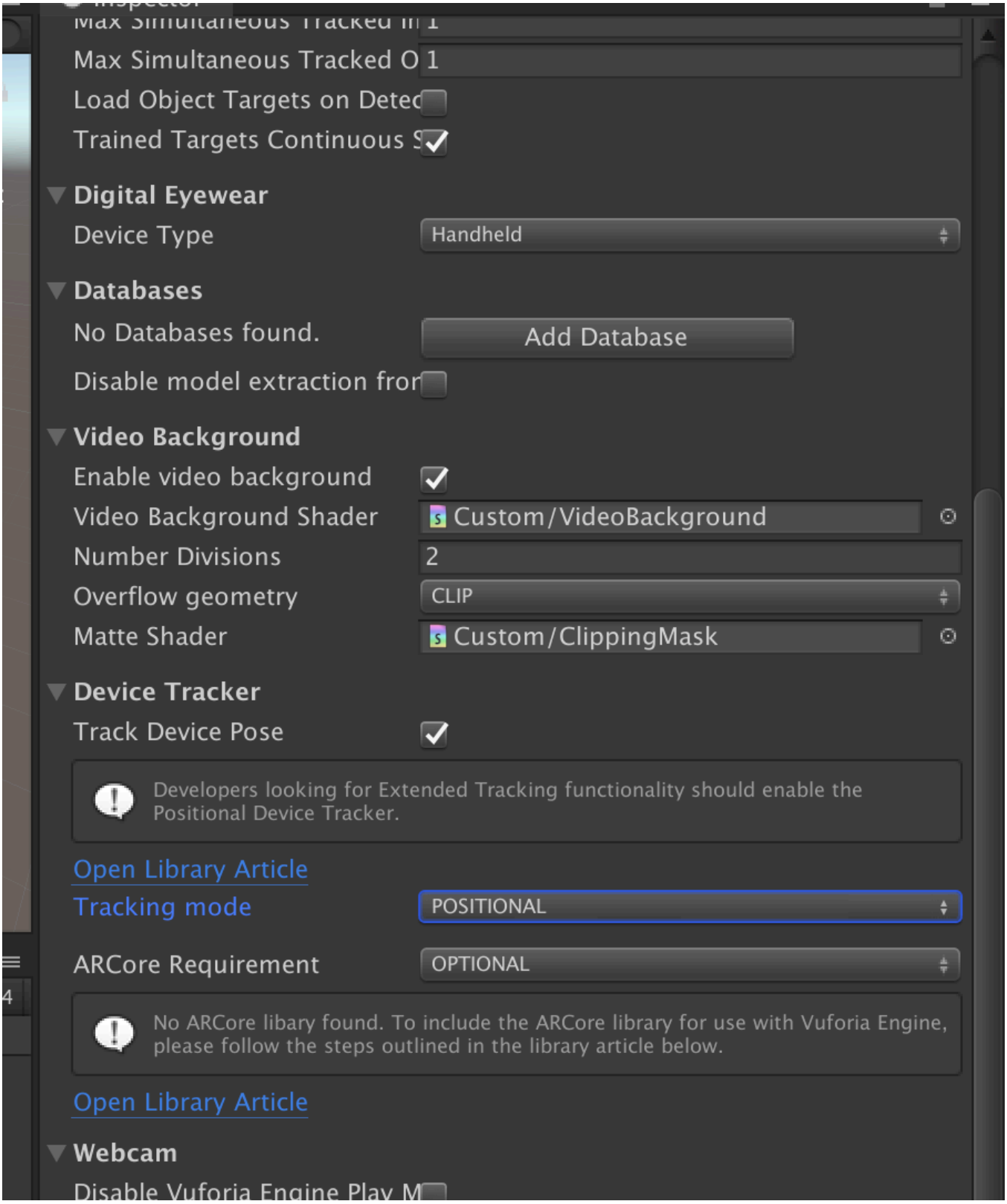

This will automatically populate the scene with all the Game Objects and Components for a basic AR+GPS experience!



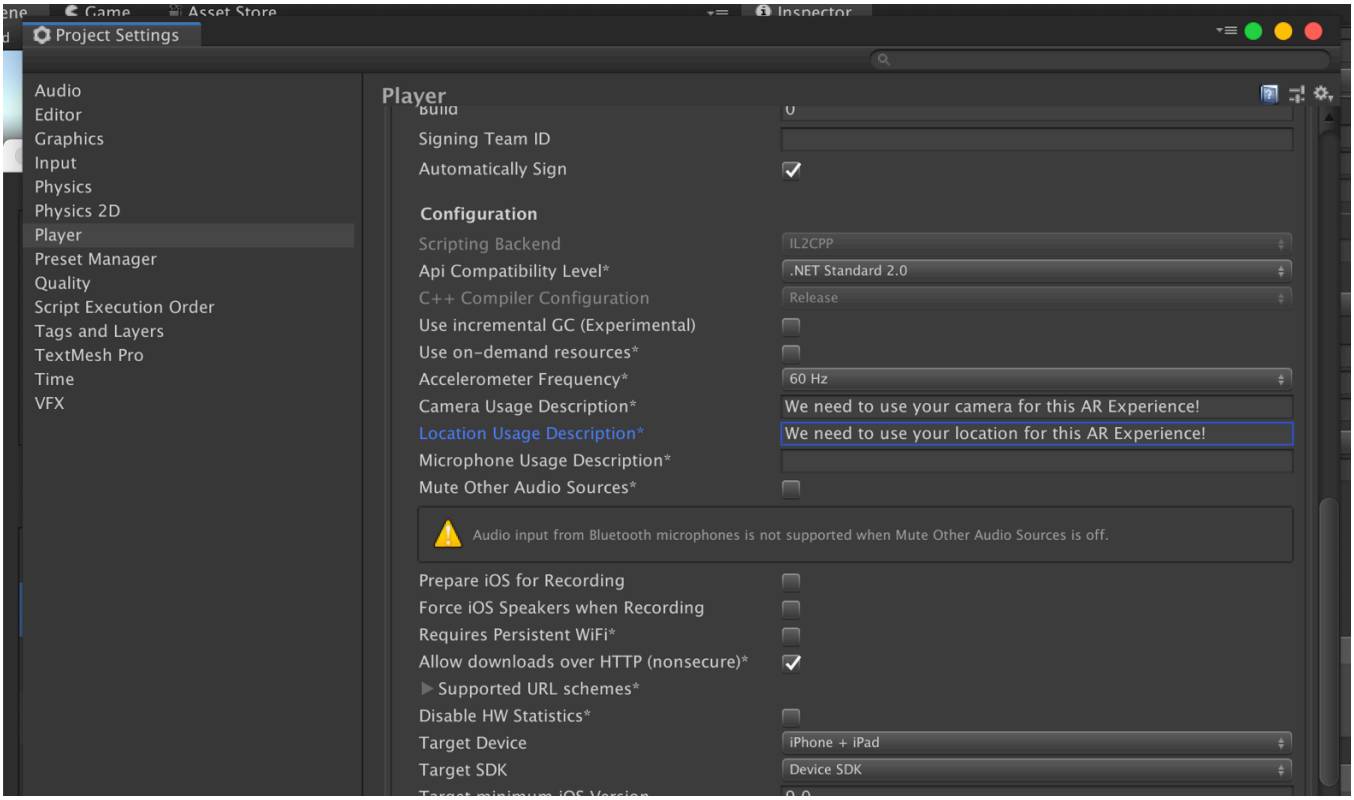
Now just click on the `GPS Stage Object` and on the `Inspector` window, go into `Location Options -> Location`, and put the geographical coordinates of the location you want the object to appear at.




Go to `Resources -> VuforiaConfiguration` and make sure that `Device Tracker -> Track Device Pose` is checked and that `Tracking Mode` is set to `POSITIONAL`. It also recommended that you enable `ARCore` support by following the instructions here, and setting `ARCore Requirement` to `OPTIONAL` in the configuration.




That's about it! Now you just need to build the project. On iOS just remember to set the `Camera Usage Description` and `Location Usage Description` strings on the Player Settings.




Now build and deploy the project onto your device, and you should the the object at the location you placed it at!

# # Basic scene structure

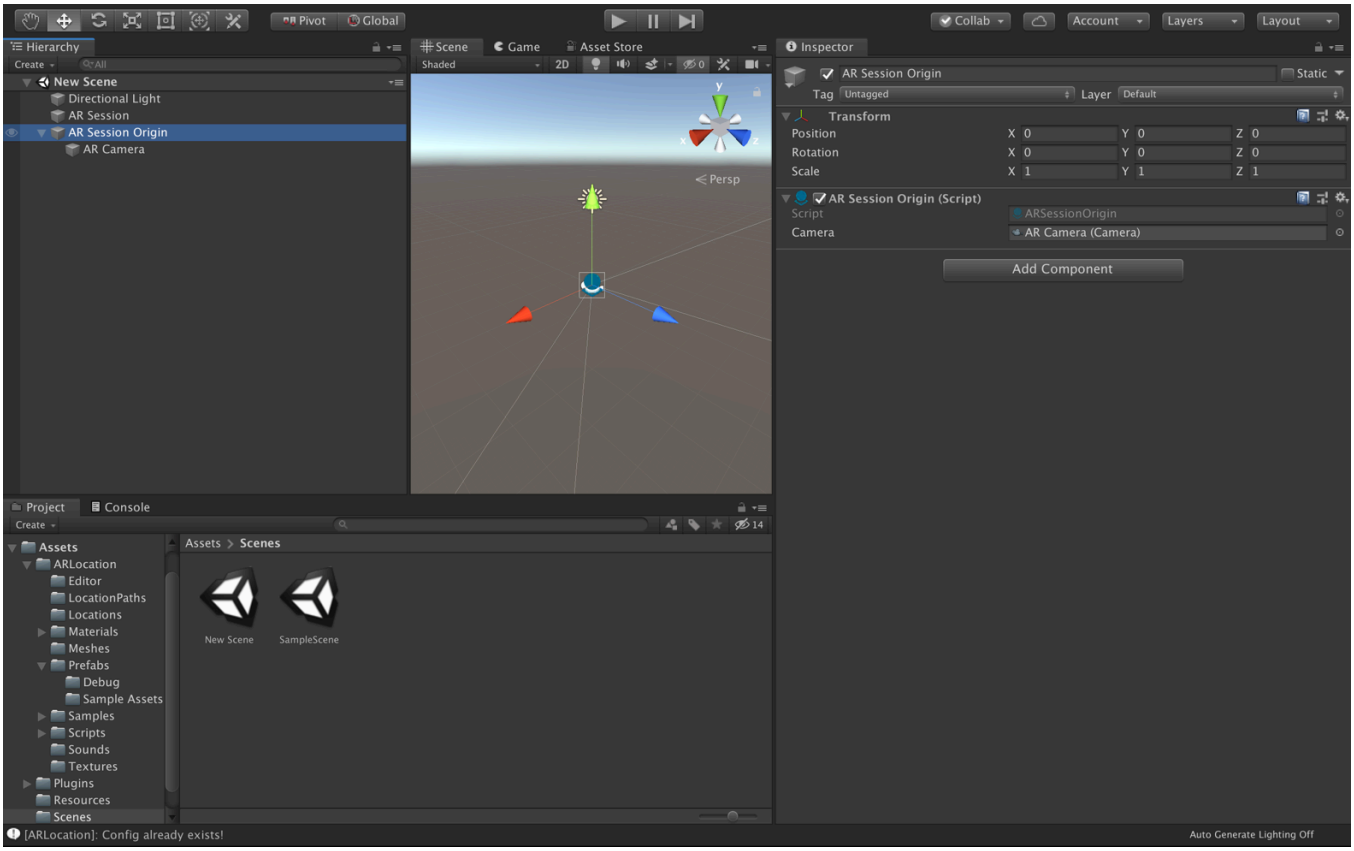
## # AR Foundation

To create a basic scene structure manually, you need to first create the basic AR Foundation game objects by going to the `Game Object` menu (i.e., right-click on the `Hierarchy` window) on clicking on the following items:

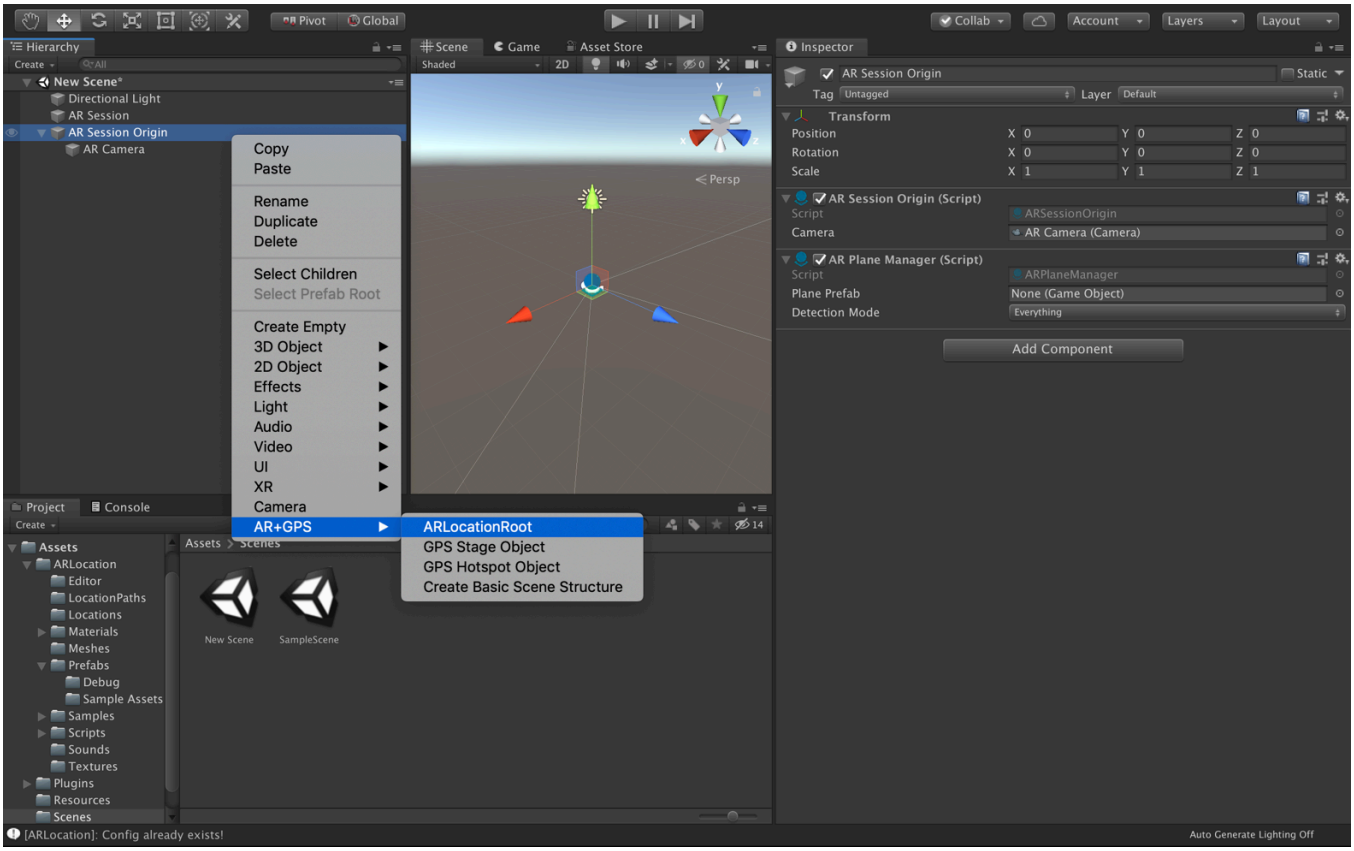
- `XR -> AR Session`
- `XR -> AR Session Origin`

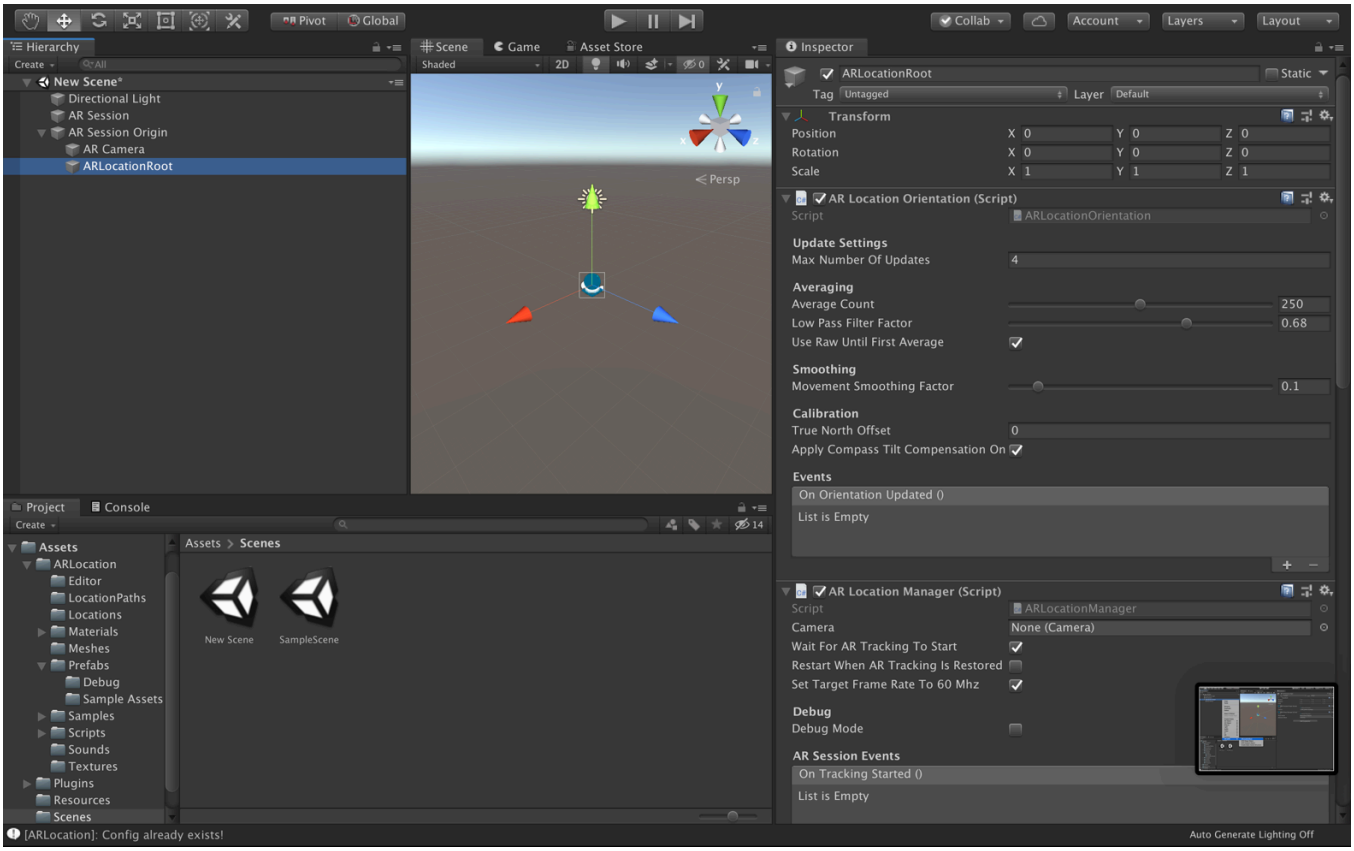
Resulting in the following scene structure:

Next, click on the `AR Session Origin` object, and add a `ARPlaneManager` component to it.

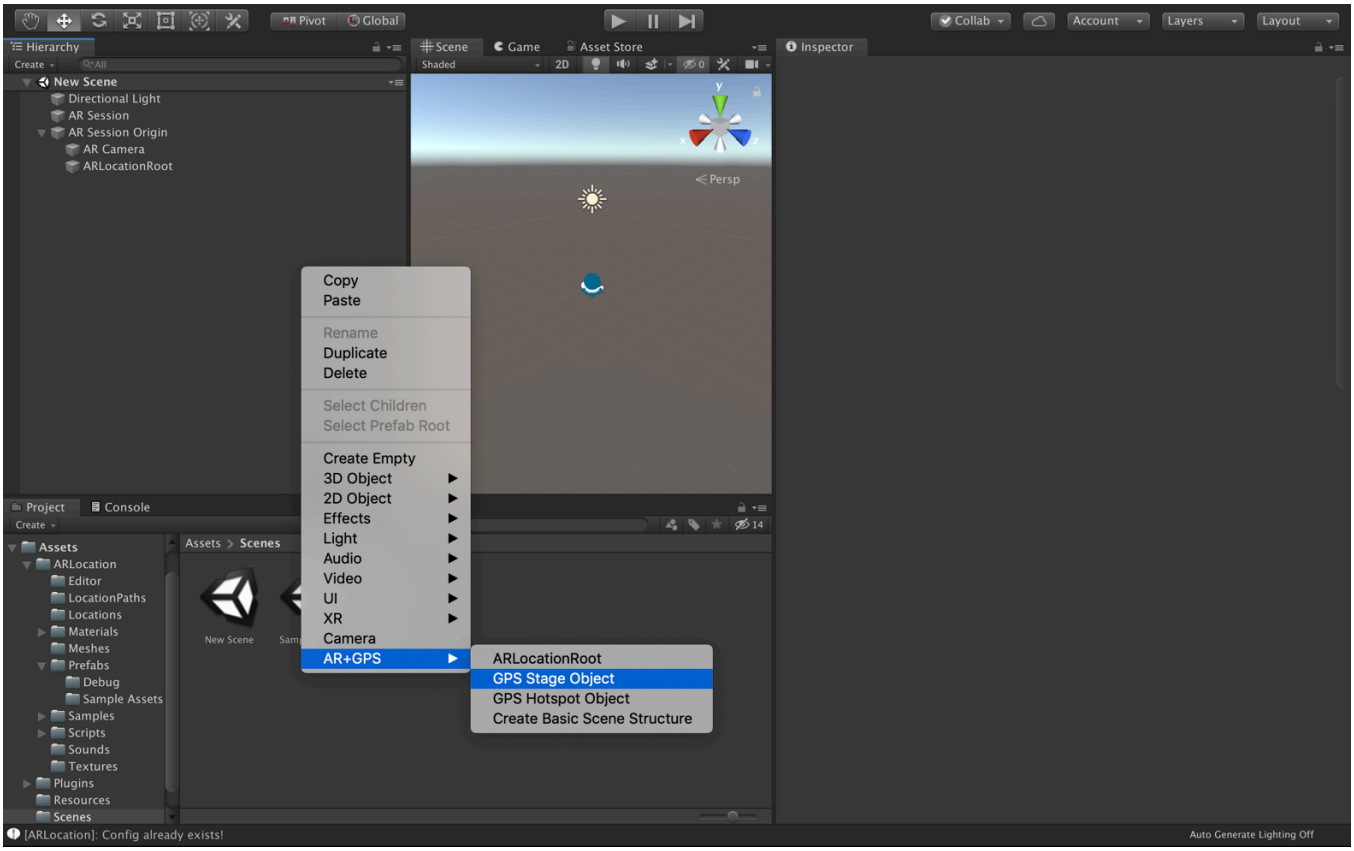



Then create a `AR+GPS -> ARLocationRoot` object as a child of the `AR Session Origin`.

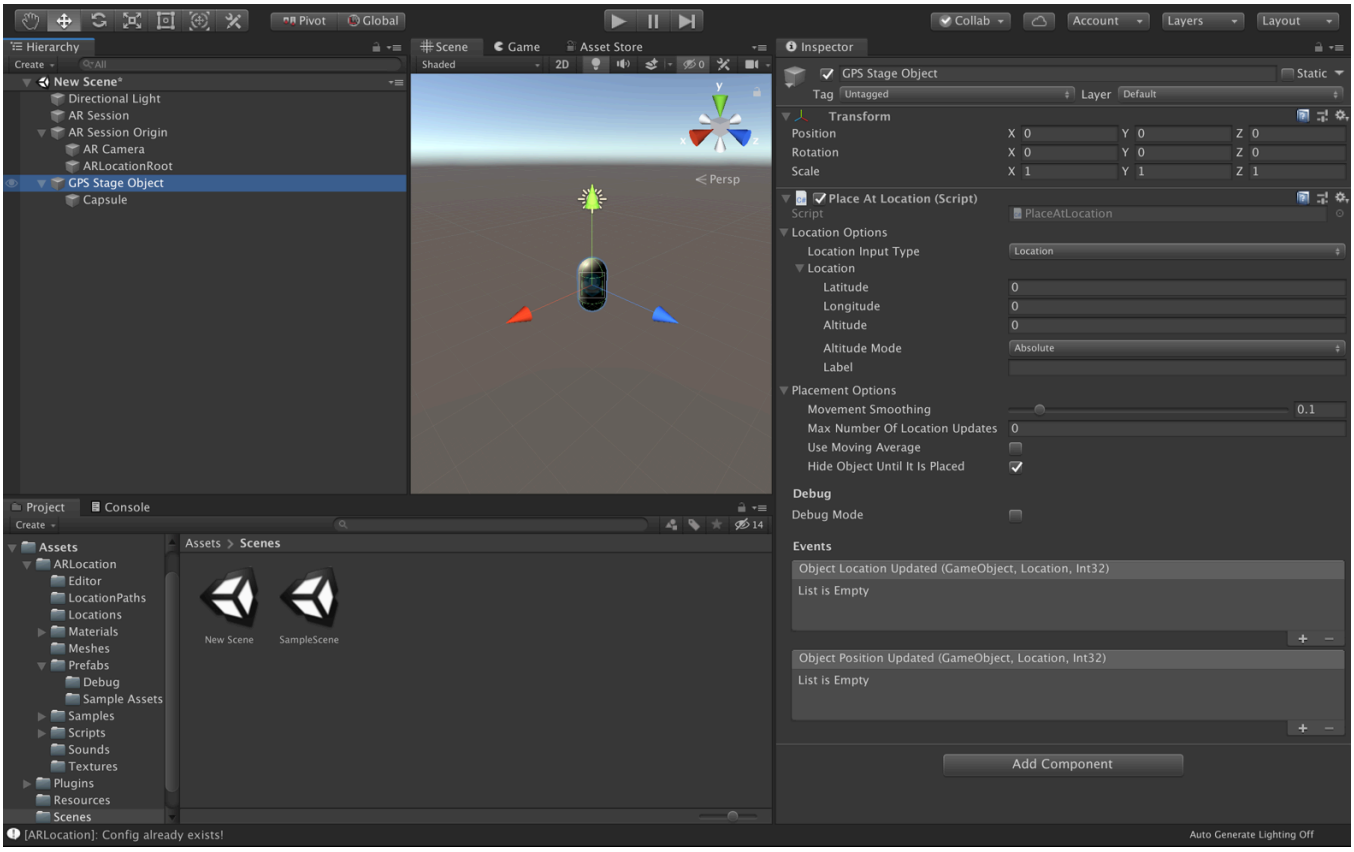





Ok, now the basic `AR+GPS` structure is laid out! All we need now is an object to be positioned in a given geo-location. There are a few components that will do that in different ways, as we will see later. For now, let's create a `AR+GPS -> GPS Stage Object`.



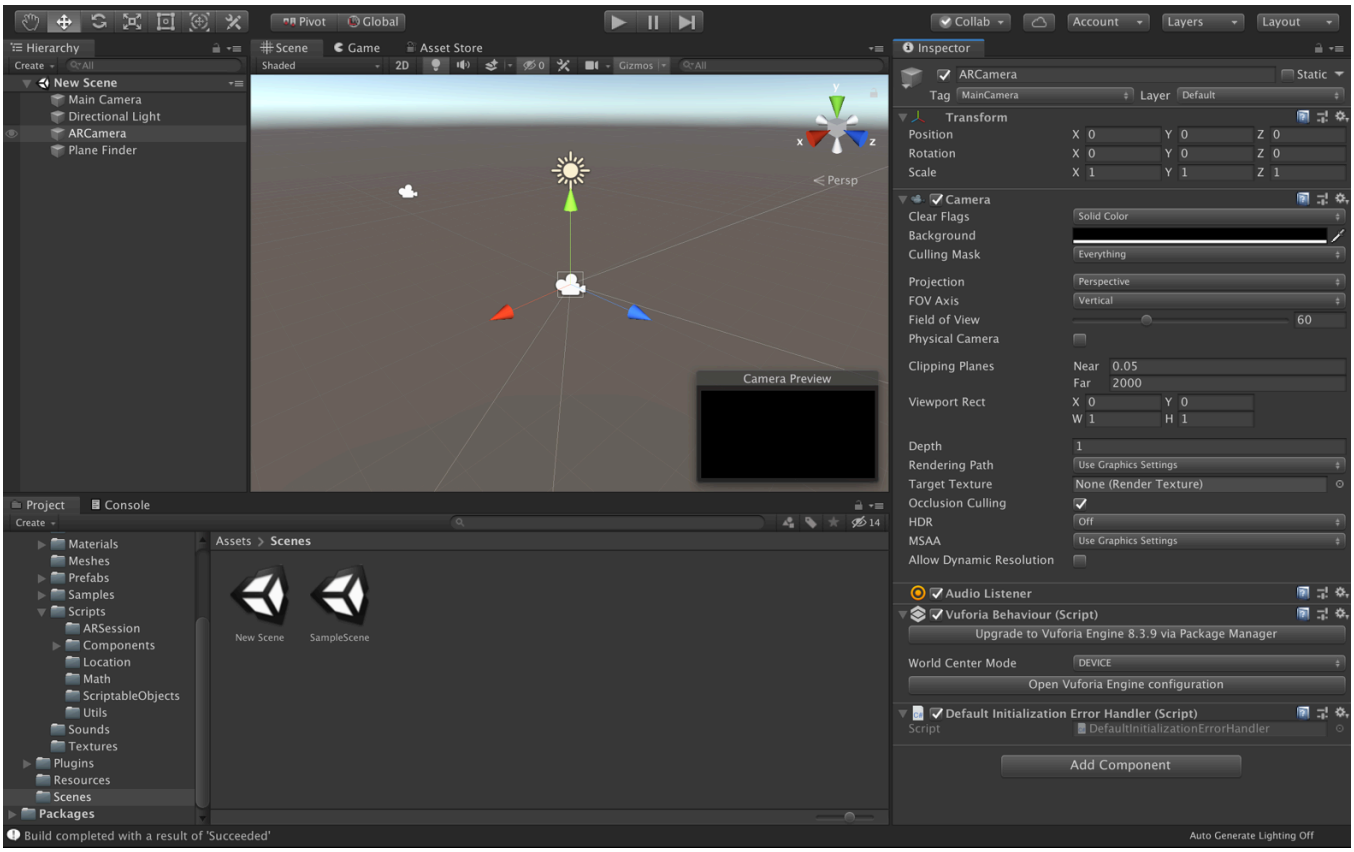

This creates an empty GameObject with a `PlaceAtLocation` component added to it. So, for something to actually appear, we should, for instance, child it to this component. Here we just create a `Capsule` as a child.



Again, all is needed is to set the desired geographical coordinates in the inspector for the `PlaceAtLocation` component of the `GPS Stage Object` to place the object in the correspond location!

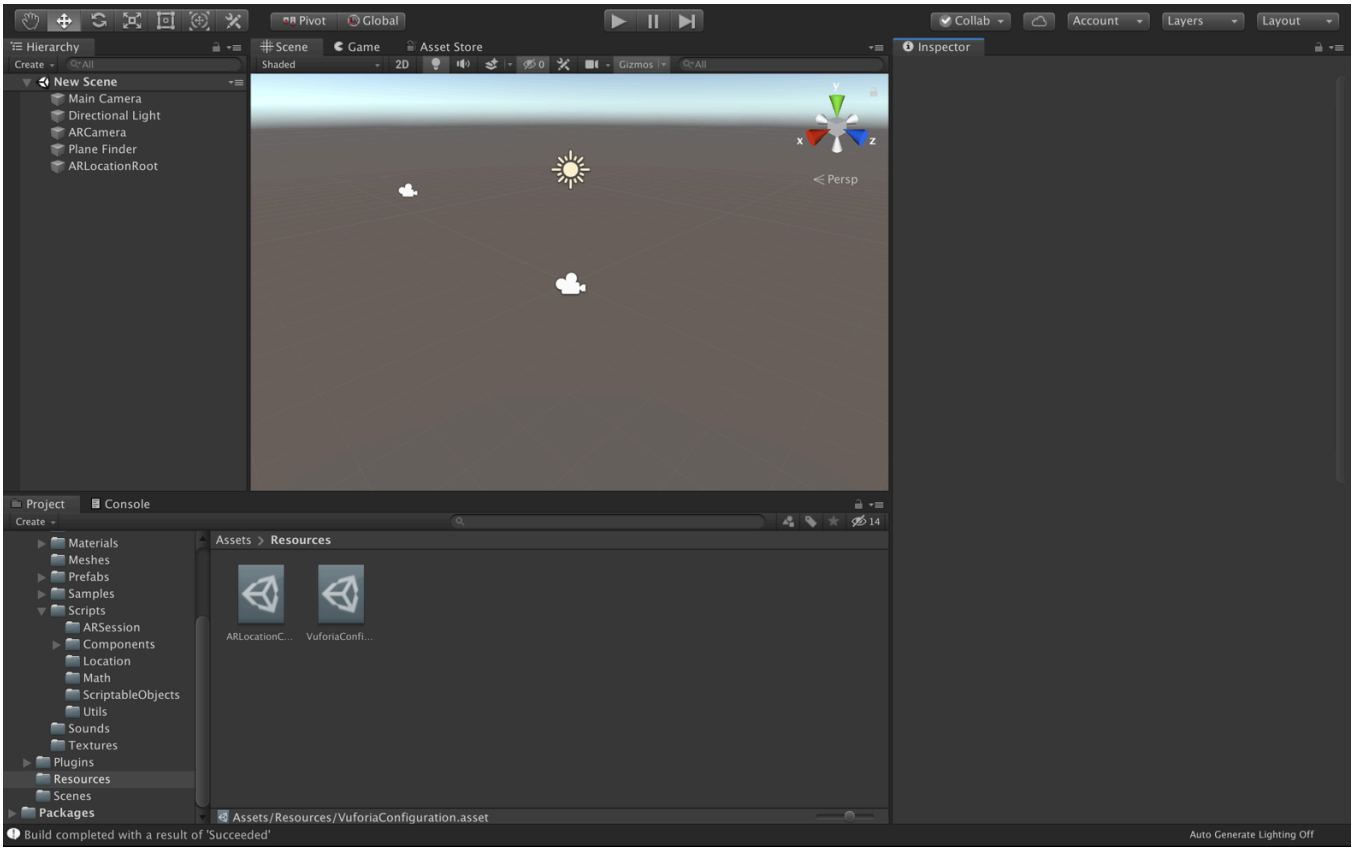
## # Vuforia

Again, starting with a new scene, delete the old `Main Camera`. Then, create a `Vuforia Engine -> AR Camera` and a `Vuforia Engine -> Ground Plane -> Plane Finder`, to create the basic Vuforia structure, like bellow.

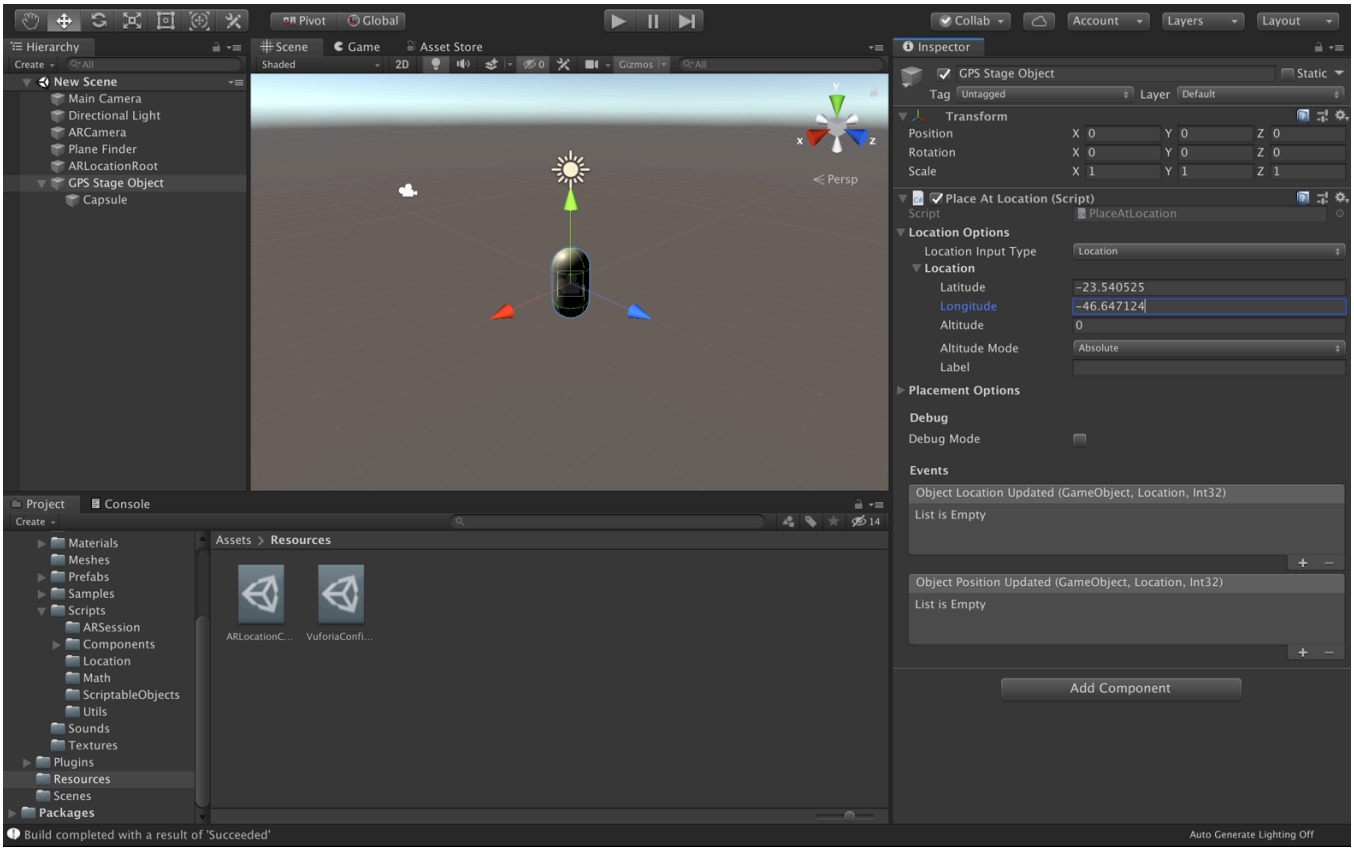



As mentioned in the last section, check that Vuforia is properly configured for positional tracking.

Now, create a `AR+GPS -> ARLocationRoot` object at the root of the scene, and the basic structure is complete.




You can add a `AR+GPS -> GPS Stage Object` and add some renderable object as it's child, setting the desired geographical coordinates at the `PlaceAtLocation` inspector for the `GPS Stage Object`.



# # Concepts

The AR+GPS system works by mixing the incoming GPS data from the device with the underlying AR tracking done by AR Foundation of Vuforia.

Both in AR Foundation and Vuforia, the AR tracking moves the camera (which starts at the origin of the world coordinates) around so that it matches the real movement of the device.

What the AR+GPS system does it to calculate the position and orientation of the gps-positioned objects relative to the user/camera, both in terms of position and orientation.

For that, it is essential that at `ARLocationRoot` object exists in the scene, as a sibling of the `AR Camera` used by the AR Tracker. Hence, in the case of AR Foundation, it will be a child of the `AR Session Origin`, while in the case of Vuforia it will be placed at the root of the scene.

The `ARLocationRoot` object must contain both a `ARLocationManager` and a `ARLocationOrientation` components. These components will ensure that the `ARLocationRoot` game object is aligned with the geographical directions, so that the objects are positioned correctly.

The `ARLocationProvider` handles all the GPS data incoming from the device. It can be placed anywhere in the scene. This data is filtered via the used configuration used by the `ARLocationProvider`.

These components are the foundation of the AR+GPS system. All the components that will actually place the objects in the scene will listen to events emitted by them to perform all the updates.

## # Altitude

Since the altitude data from mobile devices is usually unreliable, there are

a few different ways of indicating altitude/height in the `Unity AR+GPS Location` plugin. The `Altitude` property in all locations will be interpreted differently depending on the mode that is chosen.

- `Absolute` Use device's altitude data, and consider the `Altitude` as absolute, relative to the sea-level.
- `DeviceRelative` Considers the `Altitude` as being relative the the device.
- `GroundRelative` Considers the `Altitude` as being relative the the nearest detected ground plane.
- `Ignore` Disregard altitude in the calculations.

## # Stability and Precision

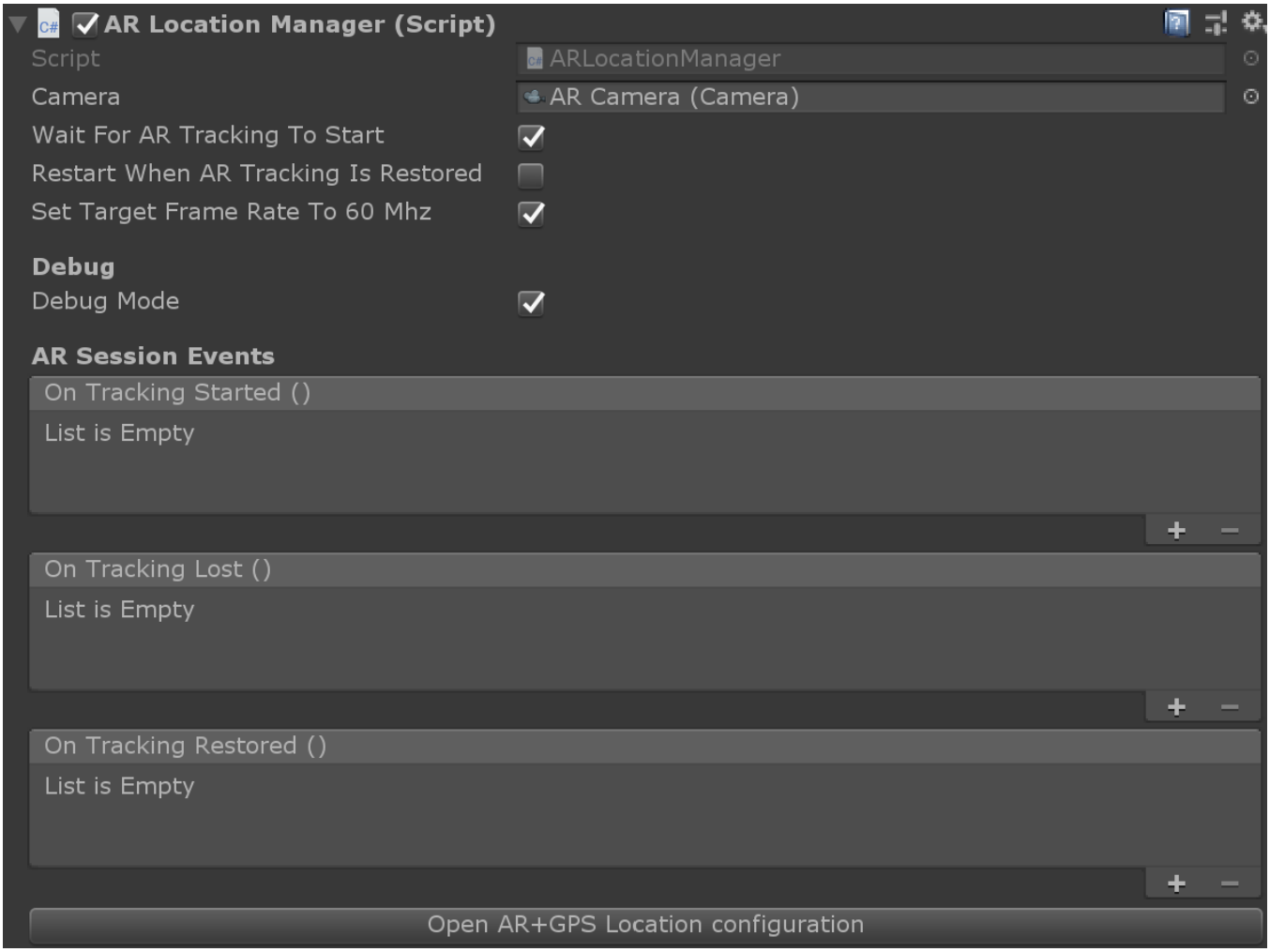
There is a certain trade-off between stability and precision; more measurements will mean, usually, better precision, but the objects will also move with each position update, disrupting stability and the persistence of the experience.

There are two situations in which objects will move due to the `AR+GPS` system: whenever there is a location update, and whenever there is a orientation update.

So, to make objects appear more stable, use low values of `Max Number Of Location Updates`, either in the individual components, or in the `ARLocationProvider` and in the `ARLocationOrientation` (but not zero, since that means infinite number of updates).

# # Components

## # ARLocationManager




The `ARLocationManager` components manages the interface between the `AR+GPS` plugin and the underlying AR session. This component is required for the `AR+GPS` system to function.

It should be placed in the `ARLocationRoot` object, which will be either a child of the `AR Session Origin` (AR Foundation) or at the root of the scene (Vuforia).

**Properties:**

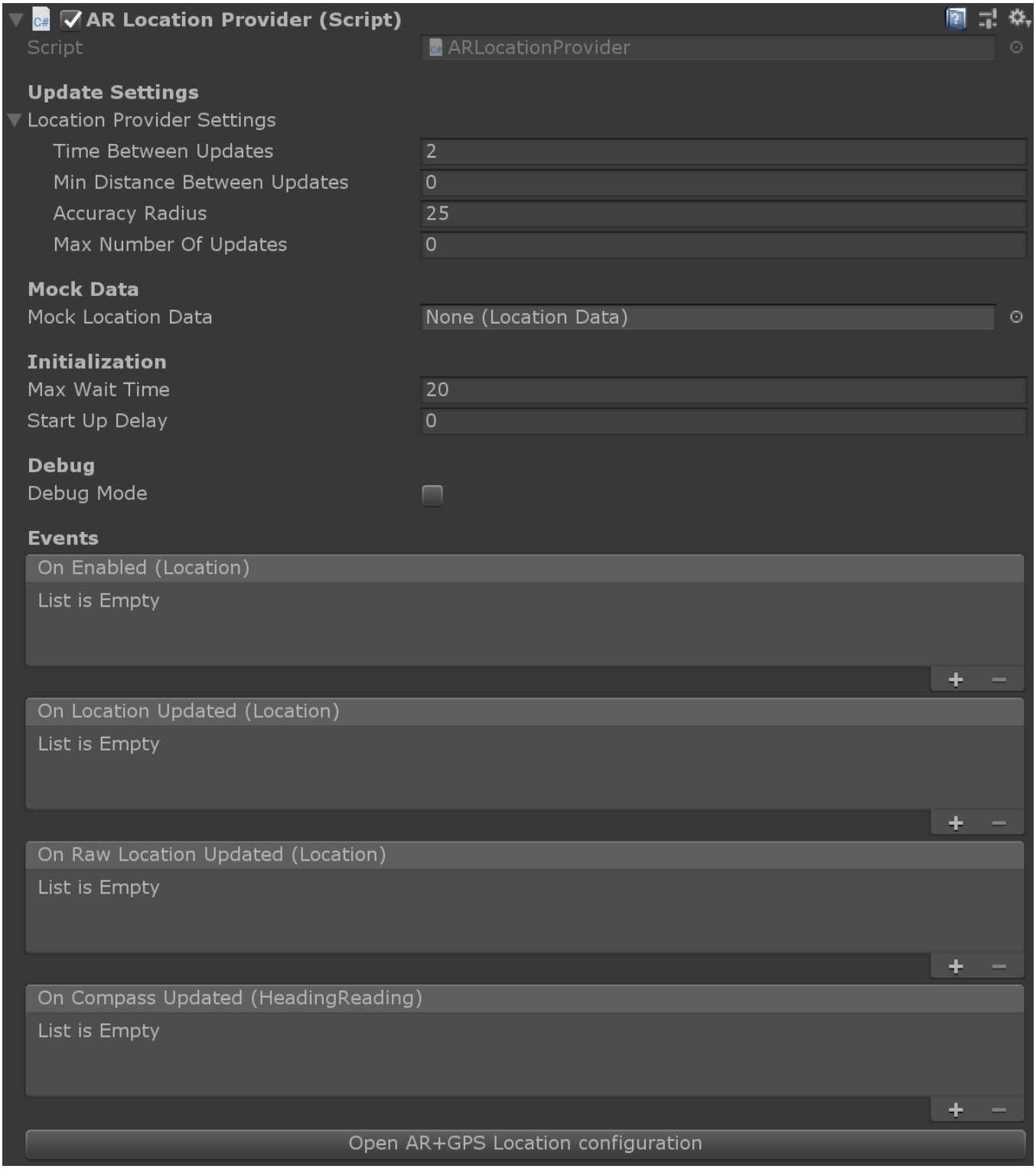
- `Camera` The AR Camera that will be used for rendering the AR content. If this is not set, the camera tagged as 'MainCamera' will be used.
- `Wait For AR Traking To Start` If true, wait until the AR Tracking starts to start with location and orientation updates and object placement. Recommended.

- `Restart When AR Tracking Is Restored` If true, every time the AR tracking is lost and regained, the AR+GPS system is restarted, repositioning all the objects.
- `Set Target Frame Rate To 60Mhz` If true, the manager will set 'Application.targetFrameRate' to 60.
- `Debug Mode` When debug mode is enabled, this component will print relevant messages to the console. Filter by 'ARLocationManager' in the log output to see the messages.

### Events

- `On Tracking Started` Emitted when the AR tracking starts for the first time.
- `On Tracking Lost` Emitted whenever the AR tracking is lost.
- `On Tracking Restored` Emitted whenever the AR tracking is restored after being lost.

## # ARLocationProvider




This component manages all the data incoming from the GPS and magnetic sensors. The data is filtered according to the settings used, but you can also have access to the raw data. This component is required for the AR+GPS system to function.

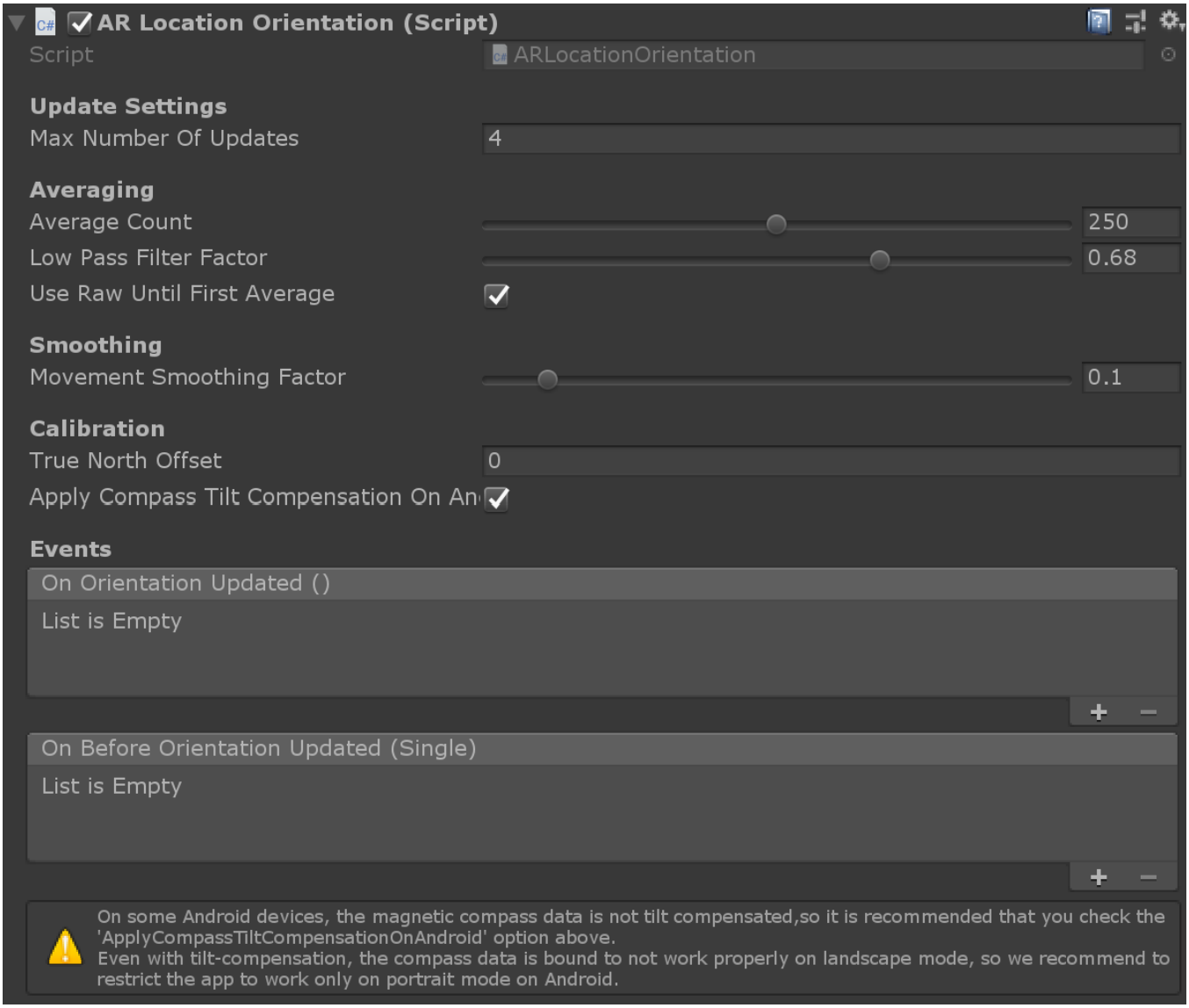
## Properties:

- `Time Between Updates` The minimum time between consecutive location updates.
- `Min Distance Between Updates` The minimum distance that will trigger location updates.
- `Accuracy Radius` The desired accuracy radius. For instance, setting this to 25 will block any data with error radius larger than 25 meters.
- `Max Number Of Updates` The global maximum number of location updates. Setting to low numbers will help block moving/repositioning of objects. Can also be set individually for each component.
- `Mock Location Data` The mock location used in-editor.
- `Max Wait Time` The maximum time to wait for location to be enabled on start-up.
- `Start Up Delay` Sets a delay for the location provider to start requesting updates. Useful for when using Unity Remote.
- `Debug Mode` When debug mode is enabled, this component will print relevant messages to the console. Filter by 'ARLocationProvider' in the log output to see the messages.

**Events:**

- `On Enabled` Emitted when the first location reading is received.
- `On Location Updated` Emitted whenever a new (filtered) location data is received.
- `On Raw Location Updated` Emitted whenever a new (raw) location data is received.
- `On Compass Updated` Emitted whenever a new compass/heading data is received.

## # ARLocationOrientation




This component manages the orientation of the `ARLocationRoot` Game Object, so that that it is in synch with the geographical directions, by using the data from the device's magnetic sensors. It should be placed on the `ARLocationRoot` Game Object, and is required for the `AR+GPS` system to function.

### Properties:

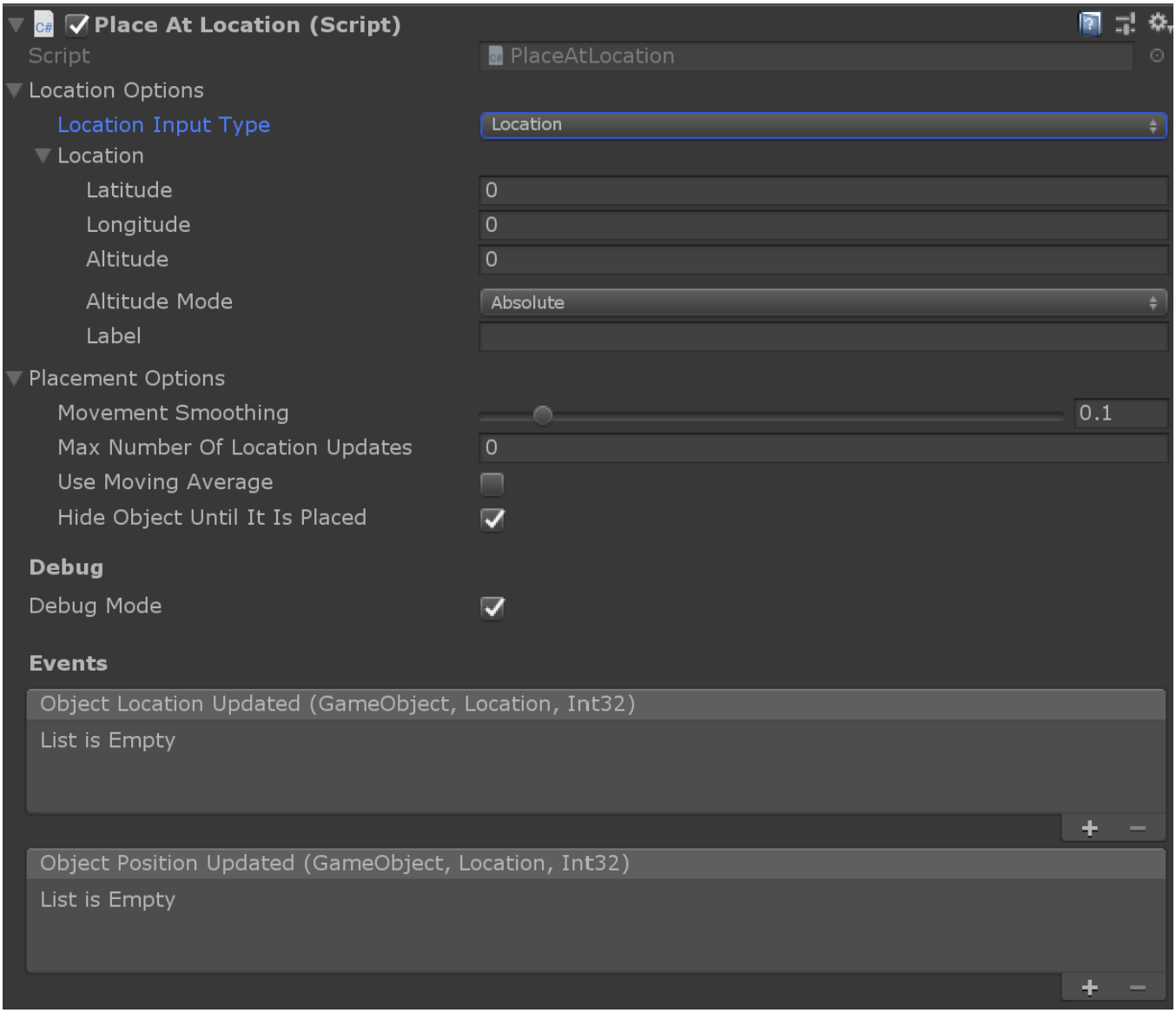
- `Max Number Of Updates` The maximum number of orientation updates. The updates will be paused after this amount. Zero means there is no limit and the updates won't be paused automatically.
- `Average Count` Only update after measuring the heading N times, and take the average.

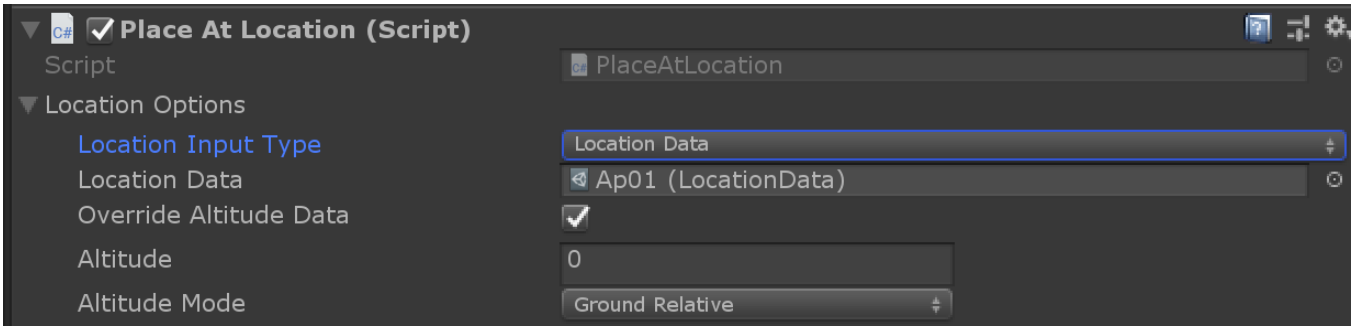
- `Low Pass Filter Factor` This is the low pass filter factor applied to the heading values to reduce jitter. A zero value disables the low-pass filter, while a value of 1 will make the filter block all value changes.
- `Use Raw Until First Average` If set to true, use raw heading values until measuring the first average.
- `Movement Smoothing Factor` The smoothing factor on the orientation rotation on change. Zero means disabled.
- `True North Offset` A custom offset to the device-calculated true north direction. When set to a value other than zero, the device's true north will be ignored, and replaced by the magnetic heading added to this offset.
- `Apply Compass Tilt Compensation On Android` If true, apply a tilt-compensation algorithm on Android devices.

**Events:**

- `On Orientation Updated` Emitted after the orientation has been updated.
- `On Before Orientation Updated` Emitted just before the orientation has been updated, with the new `y` Euler angle as argument.

# # PlaceAtLocation




This component will place the Game Object it is attached to at a given geographical location. This is the easiest way to place gps-positioned AR content.

## Properties:

- `Location Input Type` The type of location coordinate input used.

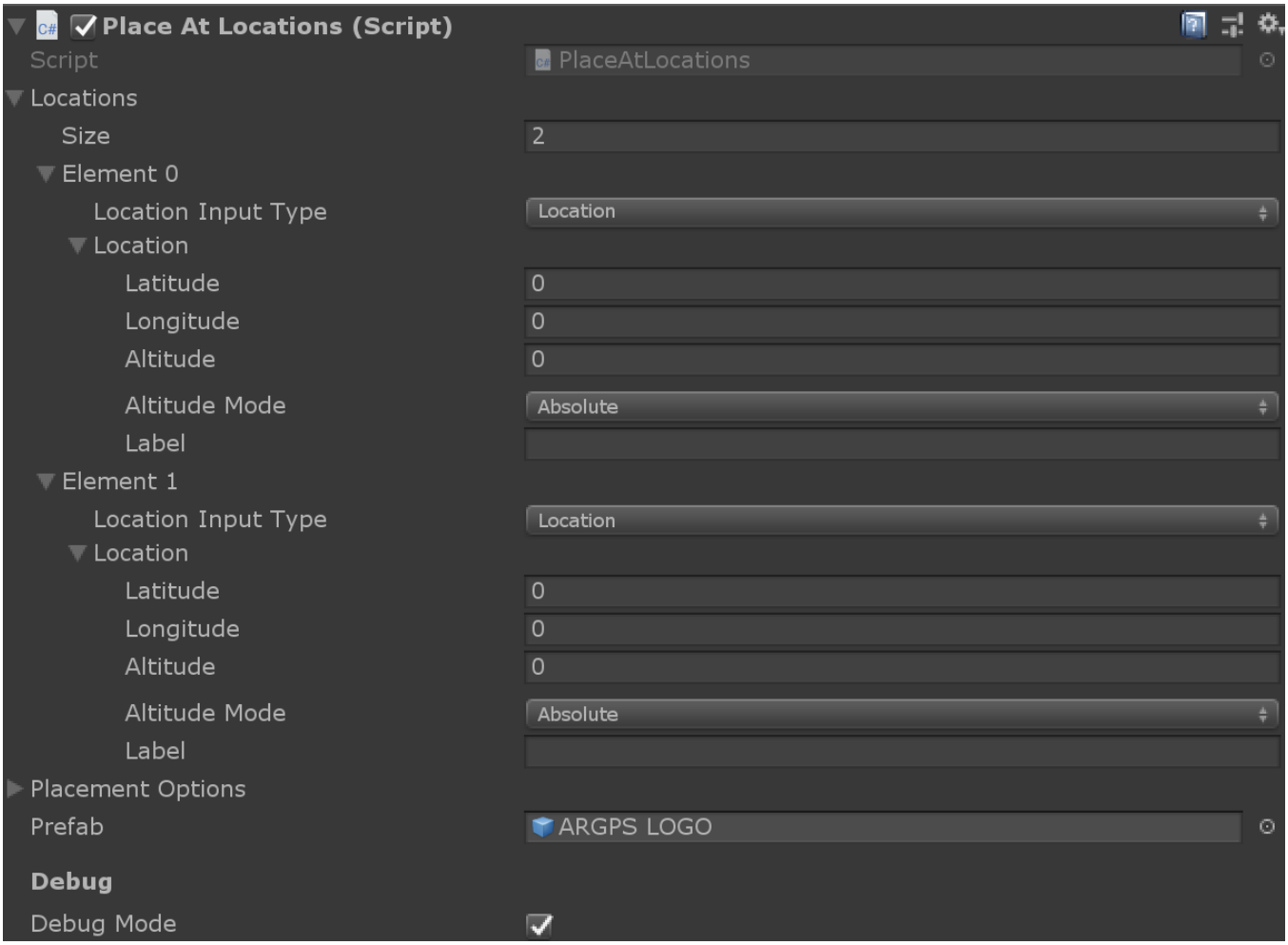
Either 'Location' to directly input location coordinates, or 'LocationData' to use a ScriptableObject.

- `Location` Input the desired GPS coordinates here.
- `Location Data` A LocationData ScriptableObject storing the desired GPS coordinates to place the object.
- `Override Altitude Data` If true, override the LocationData's altitude.
- `Movement Smoothing` The smoothing factor for movement due to GPS location adjustments; if set to zero it is disabled.
- `Max Number Of Location Updates` The maximum number of times this object will be affected by GPS location updates. Zero means no limits are imposed.
- `Use Moving Average` If true, use a moving average filter.
- `Hide Object Until It Is Placed` If true, the object will be hidden until the object is placed at the geolocation. If will enable/disable the MeshRenderer or SkinnedMeshRenderer when available, and enable/disable all child game objects.

**Events:**

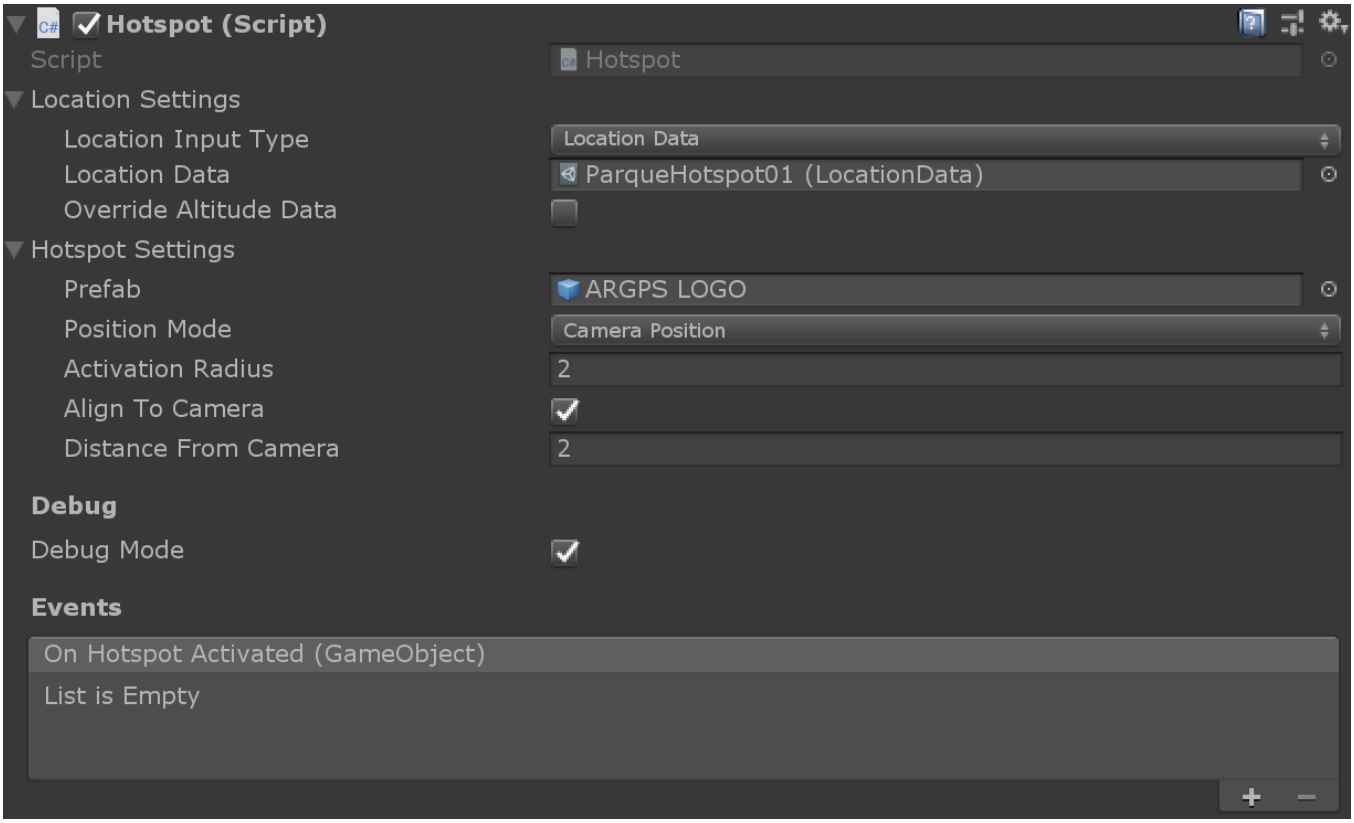
- `Object Location Updated` Event called when the object's location is updated. The arguments are the current GameObject, the location, and the number of location updates received by the object so far.
- `Object Position Updated` Event called when the object's position is updated after a location update. If the Movement Smoothing is larger than 0, this will fire at a later time than the Location Updated event. The arguments are the current GameObject, the location, and the number of position updates received by the object so far.

## # PlaceAtLocations




This component will take an array of `Locations` and instantiate a given `Prefab`, placing the instances in the given locations.

## # Hotspot




This component will create a **AR Hotspot**. Given a geo-location, when the user is inside the `Activation Radius` from that point, the hotspot will be activated, and the `Prefab` will be instantiated, either in front of the camera, at a given `Distance From de Camera`, or at the center of the Hotspot.

## Properties:

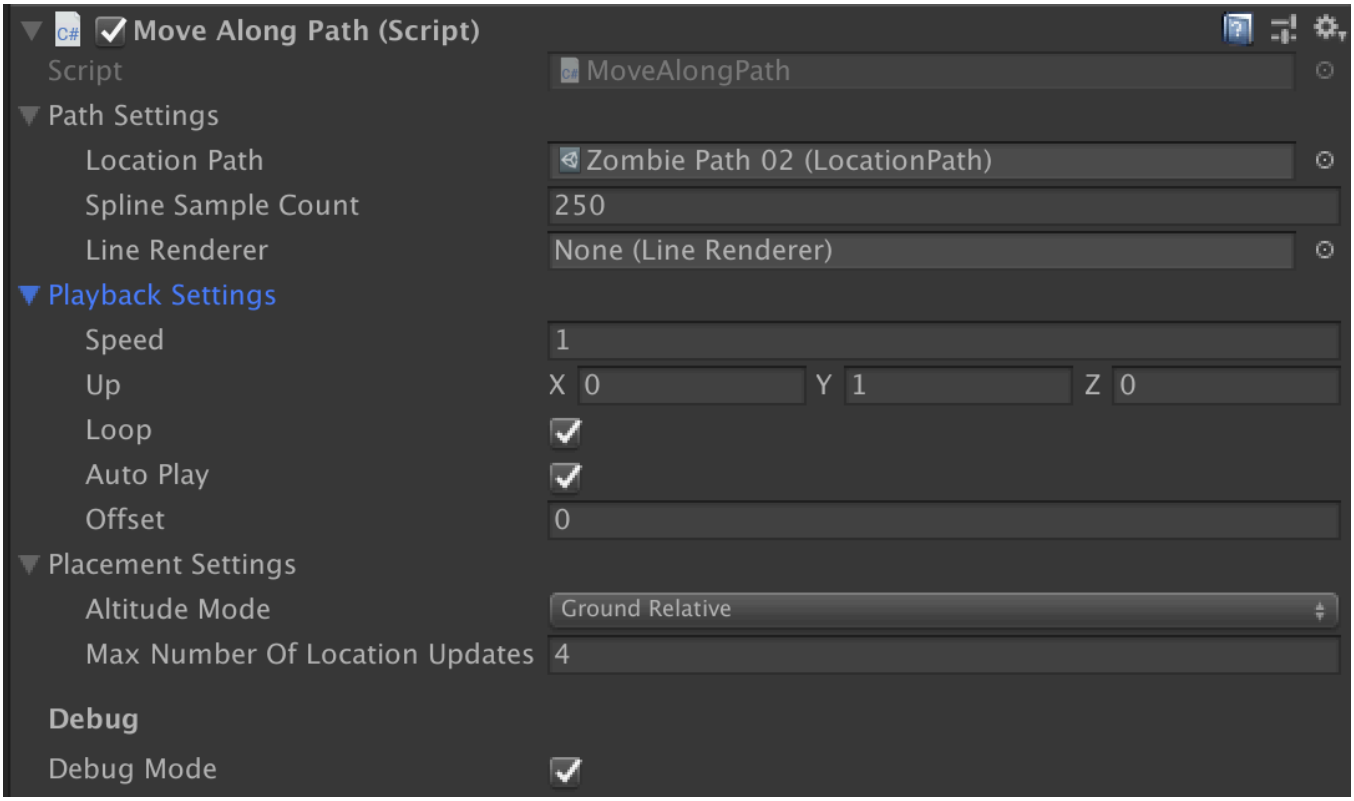
- `Prefab` The prefab/GameObject that will be instantiated by the Hotspot.
- `Position Mode` The positioning mode. 'HotspotCenter' means the object will be instantiated at the Hotpot's center geo-location. 'CameraPosition' means it will be positioned at the front of the camera.
- `Activation Radius` The distance from the center that the user must be located to activate the Hotspot.
- `Align To Camera` If true, align the instantiated object to face the camera (horizontally).
- `Distance From Camera` If 'PositionMode' is set to 'CameraPosition',

how far from the camera the instantiated object should be placed.

**Events:**

- `On Hotspot Activated` Emitted when the hotspot is activated

## # MoveAlongPath



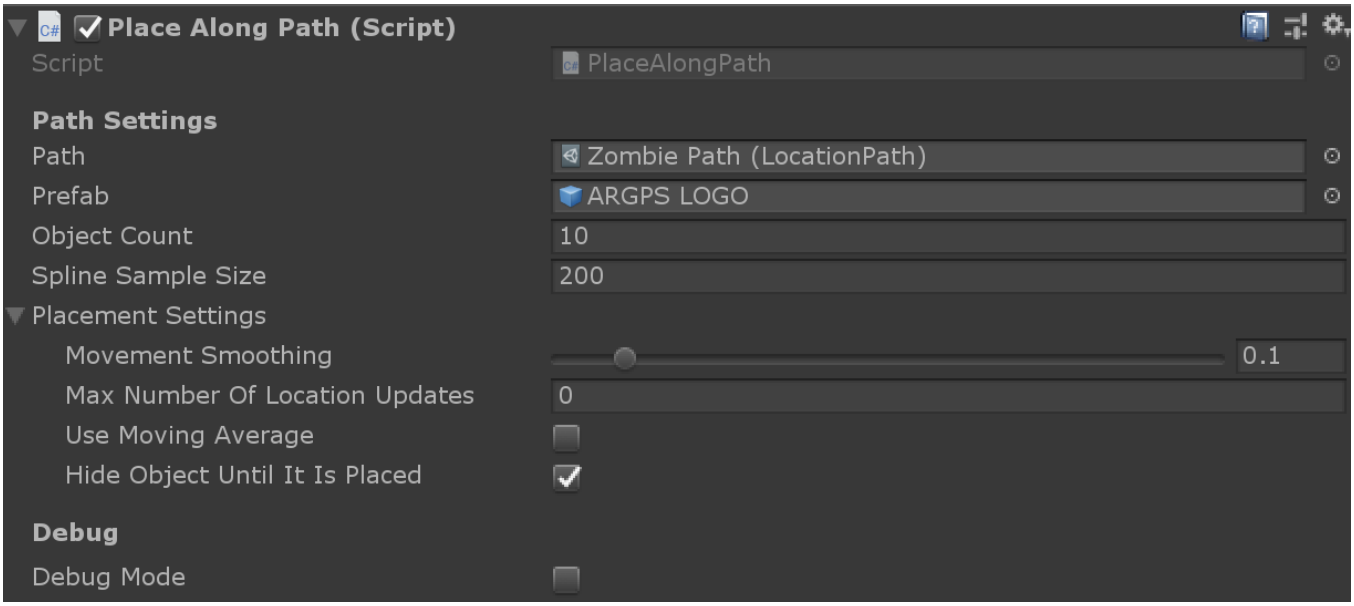

Moves a game object along a given `LocationPath`.

**Properties:**

- `Location Path` The LocationPath describing the path to be traversed.
- `Speed` The speed along the path.
- `Up` The up direction to be used for orientation along the path.
- `Loop` If true, play the path traversal in a loop.
- `Auto Play` If true, start playing automatically.
- `Spline Sample Count` The number of points-per-segment used to calculate the spline.

- `Line Renderer` If present, renders the spline in the scene using the given line renderer.
- `Offset` The parameters offset; marks the initial position of the object along the curve.
- `Altitude Mode` The altitude mode. 'Absolute' means absolute altitude, relative to the sea level. 'DeviceRelative' means it is relative to the device's initial position. 'GroundRelative' means relative to the nearest detected plane, and 'Ignore' means the altitude is ignored (equivalent to setting it to zero).
- `Max Number Of Location Updates` The maximum number of times this object will be affected by GPS location updates. Zero means no limits are imposed.

## # PlaceAlongPath



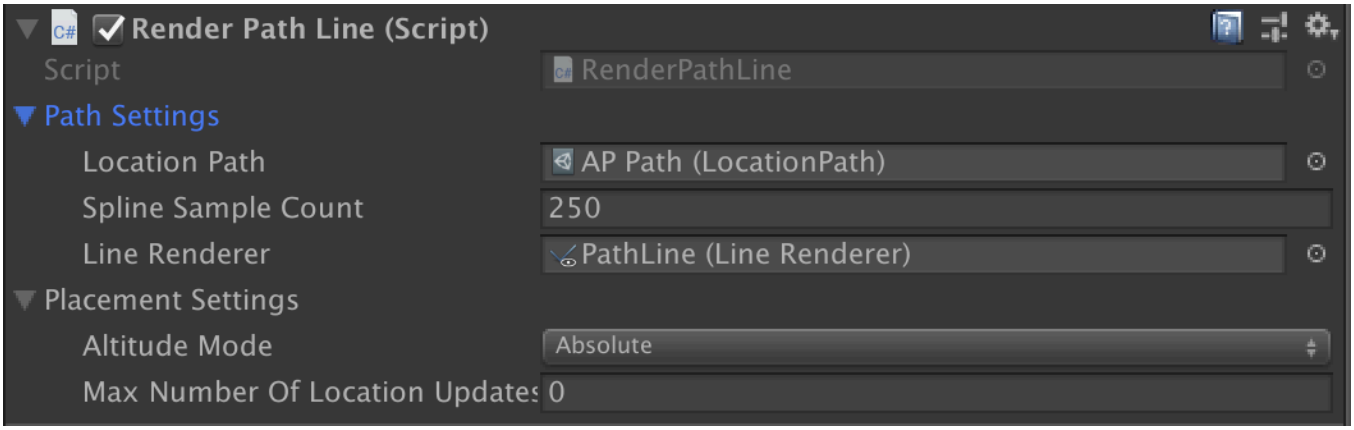

Instantiates the given `Prefab`, placing the instances along the `Path`.

### Properties:

- `Path` The LocationPath describing the path where objects will be placed.
- `Prefab` The prefab to be instantiated.

- `Object Count` How many instances will be created.
- `Spline Sample Size` The number of points-per-segment used to calculate the spline.
- `Movement Smoothing` The smoothing factor for movement due to GPS location adjustments; if set to zero it is disabled.
- `Max Number Of Location Updates` The maximum number of times this object will be affected by GPS location updates. Zero means no limits are imposed.
- `Use Moving Average` If true, use a moving average filter.
- `Hide Object Until It Is Placed` If true, the object will be hidden until the object is placed at the geolocation. If will enable/disable the MeshRenderer or SkinnedMeshRenderer when available, and enable/disable all child game objects.

## # RenderPathLine




Renders the given `LocationPath` using a `LineRenderer`.

### Properties:

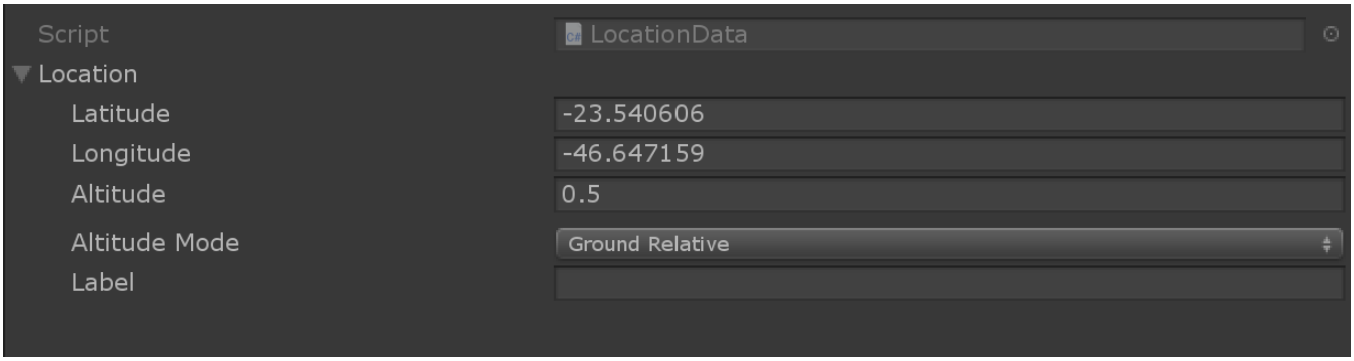
- `Location Path` The LocationPath that will be rendered.
- `Altitude Mode` The altitude mode. 'Absolute' means absolute altitude, relative to the sea level. 'DeviceRelative' means it is relative to the device's initial position. 'GroundRelative' means relative to the nearest detected plane, and 'Ignore' means the altitude is ignored

(equivalent to setting it to zero).

- `Spline Sample Count` The number of points-per-segment used to calculate the spline.
- `Line Render Sample Count` The number of points-per-segment used draw the spline.
- `Line Renderer` The LineRenderer to render the Path with.

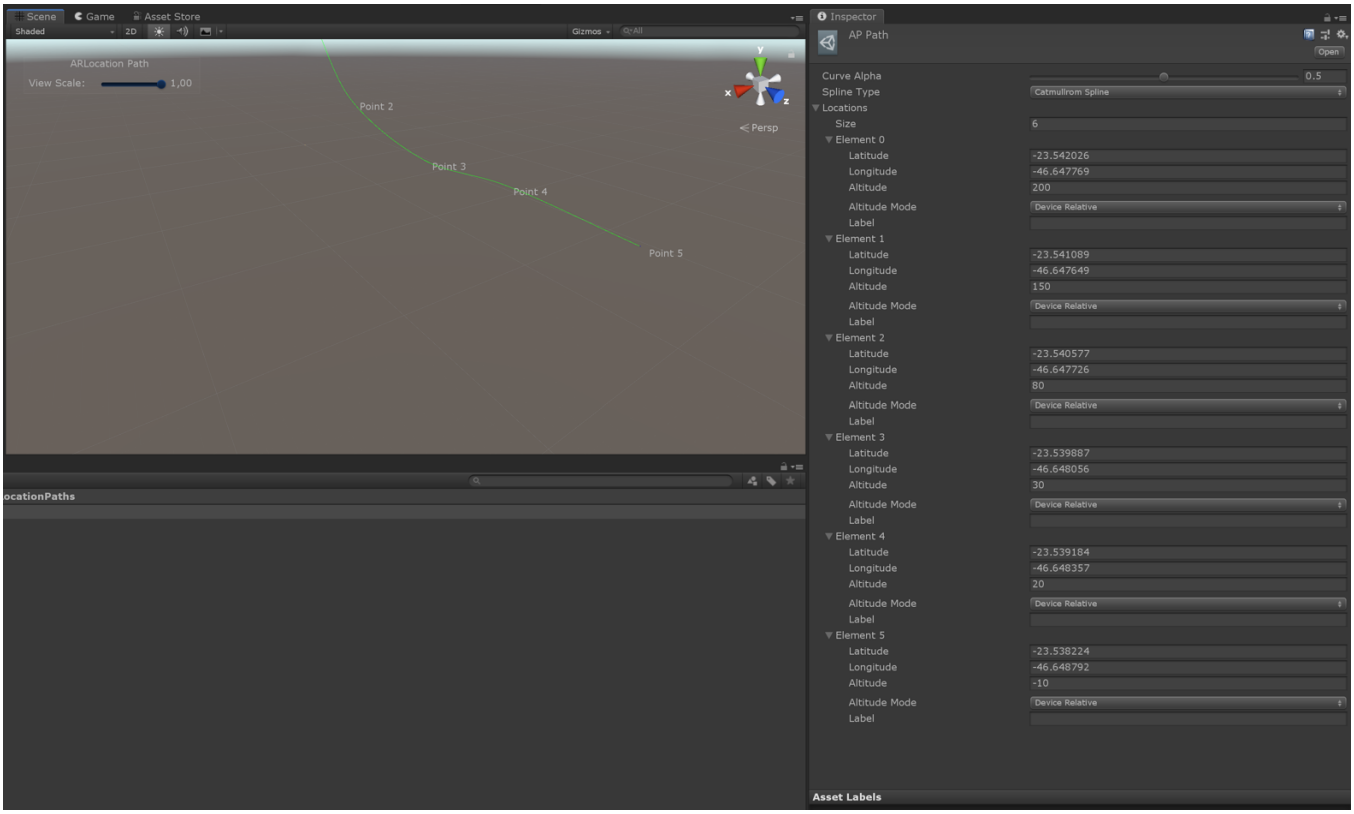
# # ScriptableObjects

## # LocationData



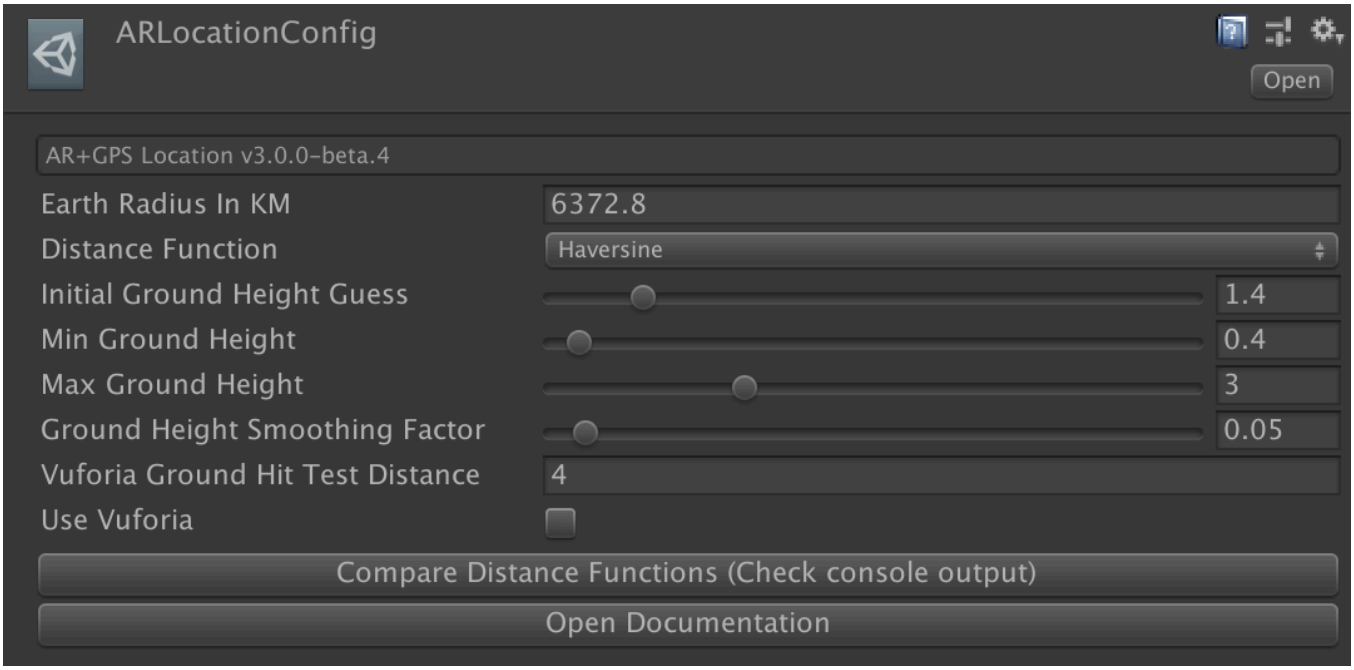

Holds the geo-coordinates for a location. To create, go to `Create -> AR+GPS -> Location`.

## # LocationPath




Holds data for a path/curve in space, defined by a set of geo-coordinates. To create, go to `Create -> AR+GPS -> Path`.

# # ARLocationConfig




This holds the global configuration for the `Unity AR+GPS Location`. The plugin will automatically create a configuration file at

`Resources/ARLocationConfig`.

**Properties:**

- `Earth Radius in KM` The value of the Earth's radius, in KM, used for distance calculations.
- `Distance Function` The distance function used to calculate geographical distances.
- `Initial Ground Height Guess` The initial guess for the distance between the device and the ground.
- `Min Ground Height` The minimum distance between the device and the ground (for when detecting the floor planes)
- `Max Ground Height` The maximum distance between the device and the ground (for when detecting the floor planes)
- `Vuforia Ground Hit Test Distance` The minimum distance, in meters, between Hit tests when detecting the nearest ground plane in Vuforia.
- `Ground Height Smoothing Factor` The movement smoothing for when the ground plane height changes.
- `Use Vuforia` Enable/disable Vuforia.

# # Scripting

## # Placing objects at runtime

Suppose you want to place a single object at a given location, at runtime.

```
var loc = new Location()
{
    Latitude = myLatValue,
    Longitude = myLngValue,
    Altitude = myAltValue,
    AltitudeMode = AltitudeMode.GroundRelative
```

```
};

var opts = new PlaceAtLocation.PlaceAtOptions()
{
    HideObjectUntilItIsPlaced = true,
    MaxNumberOfLocationUpdates = 2,
    MovementSmoothing = 0.1f,
    UseMovingAverage = false
};

PlaceAtLocation.AddPlaceAtComponent(gameObject, loc, opts);
```

You can also use the `PlaceAtLocation.CreatePlacedInstance` to instantiate a given Prefab/game object at a given location.

If you have many objects/location, simply loop trough all of them, applying the formula above.

## # Updating the object location

If you placed an object using the `PlaceAtLocation` component, you can easily update it's geographical location:

```
var newLocation = new Location()
{
    Latitude = myLatValue,
    Longitude = myLngValue,
    Altitude = myAltValue,
    AltitudeMode = AltitudeMode.GroundRelative
};

var placeAtLocation = GetComponent<PlaceAtLocation>();
placeAtLocation.Location = newLocation;
```

## # Distance from the user to a placed object

Once an object has been placed using the `PlaceAtLocation` component, you can monitor the distance from the user to the object by calling

```
var distance = placeAtComponent.SceneDistance;
```

This simply a shorthand for the distance between the game object and the camera. You can also get the raw/unfiltered gps distance by doing

```
var rawDistance = placeAtComponent.RawGpsDistance;
```

Both can be useful for triggering actions based on the distance of the user to the object.

## # Resetting

To reset both the AR tracker and the `AR+GPS` system, call

```
ARLocationManager.Instance.ResetARSession((() =>
{
    Debug.Log("AR+GPS and AR Session were restarted!");
}));
```

If you want to reset only the `AR+GPS` system, you can write

```
ARLocationManager.Instance.Restart();
```

## # Custom Location Providers

The location provider that is used by the `AR+GPS` system, is the `UnityLocationProvider` class. If, for some reason, you wish to use your own custom location provider (e.g., you want to use a native module, or

have a special type of device you want to support), just create a class that implements the `ILocationProvider` interface. Then add `ARGPS_CUSTOM_PROVIDER` to the `Scripting Define Symbols` in the Player settings, and finally, edit the script `ARLocation/Scripts/Components/ARLocationProvider` and on line 130, instantiate your provider:

```
#elif ARGPS_CUSTOM_PROVIDER


        Provider = new ARGpsCustomLocationProvider();
#else
```

## # Geographical Location From World Position

If you need the geographical coordinates for a given world-position (for instance, if you want the user to place objects around him, and get the lat/lon coordinates to save these locations to be restored in another session), use the following code:

```
Location location = ARLocationProvider.Insance.GetLocationForWorldPosition(
```

# # Troubleshooting

Some tips to avoid common errors:

- Make sure your device has a working GPS and magnetic sensors.
- When using AR Foundation, always check if ARKit and/or ARCore are installed.
- When using Vuforia, make sure `AR Foundation` package is installed but that ARKit and ARCore are not.
- Don't use landscape mode on Android, since the tilt-compensation

algorithm currently does not support it.
- Always make sure the `AR Camera` is tagged as the `MainCamera` or that it is set in the `ARLocationManager` as the `Camera` property.
- When using Vuforia, make sure positional tracking is activated.

## # Debugging

- Many components have a `Debug Mode` property that, when set, will print relevant information to the devices console. For instance, when using the `PlaceAtLocation` component, you can use this to keep track of the position changes of the object.

- The `Prefabs/Debug/ARLocationInfo` prefab contains a `Canvas` which will display useful information on the screen.

## # Reporting bugs

You can report bugs here.

You can also ask questions, submit other issues and/or feature requests in this link.

## # External Resources

- The ARFoundation manual.
- The Unity AR Handheld forums.
- The Vuforia Unity Forums.