

Google Maps View Documentation

Table of Contents

[Overview](#)

[Limitations](#)

[Setup](#)

- [1. Change the default Bundle id to your Bundle Id \(package name\)](#)
- [2. Obtaining Google API Key](#)
- [3. Put the API key inside your AndroidManifest.xml inside Unity Project.](#)
- [4. Run the Demo Scene](#)

[Usage Instructions](#)

[Creating, Showing and Dismissing GoogleMapView](#)

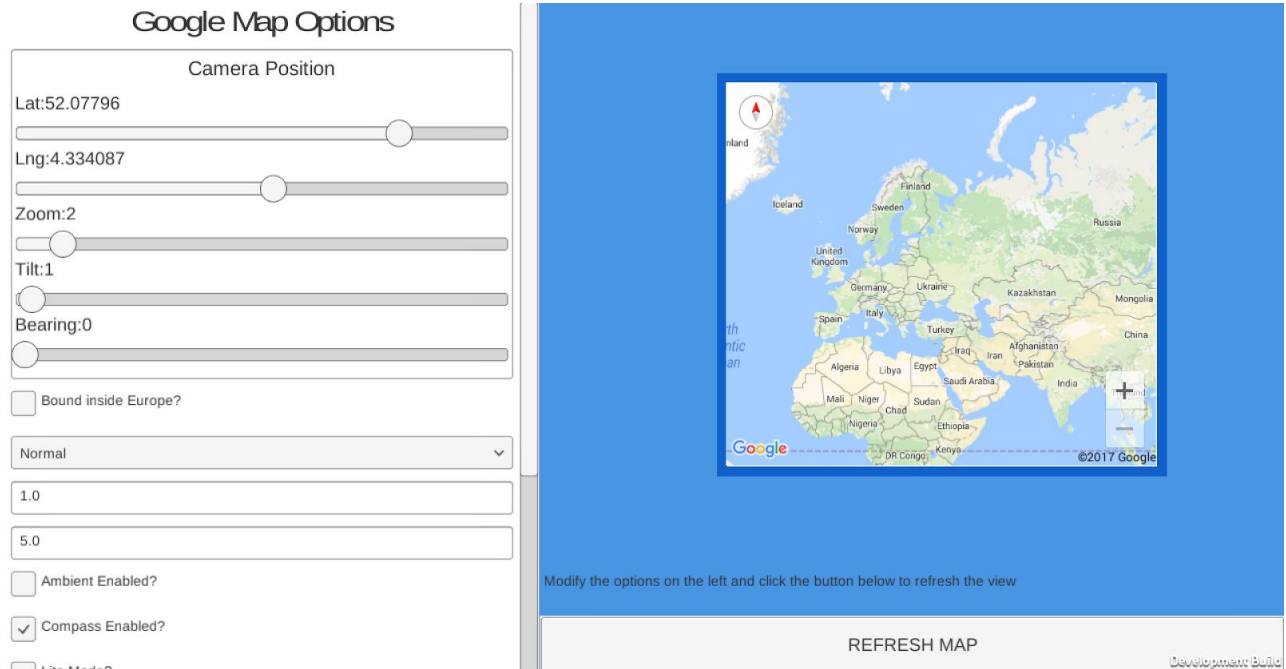
[FAQs and Known Issues](#)

For the latest up-to-date documentation always visit the [Documentation Wiki](#). It is highly recommended to use Wiki as it is always up-to-date and easier to navigate.

For support drop me a message to leskiv.taras@gmail.com

Overview

The plugin allows you to embed Native [GoogleMapView](#) into your Android-Unity game. Note, this is NOT a Web View and NOT a Texture, its native interactive Google Map View.



Limitations

These are the limitations to be aware of when using the plugin:

- The plugin does NOT work in Editor! It is a native Android view (not a web view) so performance is awesome but there is no way to get native Android view working in Unity Editor.
- You have to handle device orientation changes. Use `Screen.orientation` to determine screen orientation changes, when it changes, dismiss the view and show again passing new position rect.
- You have to handle dismissing and showing the view again in **void OnApplicationPause(bool pauseStatus)** because you might get a black screen behind map if you don't. This is very straightforward to implement (also see demo):
- You can't move the view (e.g scroll in Unity view). The view for now is static and can't be moved around. Repositioning might be implemented in future. Please submit an issue to this repo to request this.

Setup

This section describes how to get Google Maps View running in your app or demo that is included with plugin.

1. Change the default Bundle id to your Bundle Id (package name)

Go to Unity Android Player Settings and set the Bundle Id as your package name, e.g. `gmaps.deadmosquitogames.com.googlemaps` and save it. **This is important as Google API Key in the next step is bound to the package you set.** In this document I will refer to your package as `YOUR_PACKAGE_NAME`

Identification	
Bundle Identifier	<input type="text" value="gmaps.deadmosquitogames.com.googlemaps"/>
Version*	<input type="text" value="1.0"/>
Bundle Version Code	<input type="text" value="1"/>
Minimum API Level	<input type="text" value="Android 4.0 'Ice Cream Sandwich' (API level 14)"/>

2. Obtaining Google API Key

This part is a bit tricky so please follow instructions carefully.

- If you don't already have a [Google Console](#) account create one and login.
- Go to <https://developers.google.com/maps/documentation/android-api/signup> and click **GET A KEY** button.

Google Maps APIs

Home Documentation Pricing and Plans

OVERVIEW GUIDES REFERENCE SAMPLES SUPPORT

Get Started

- Project Setup
- Overview
- Configuration
- Get API Key

Tutorials

- Map with Marker
- Polylines and Polygons to Represent Routes and Areas

Creating a Map

- Map Objects
- Businesses and Other Points of Interest
- Lite Mode
- Street View
- Android Wear

Get API Key

To use the Google Maps Android API, you must register your app project on the Google API Console and get a Google API key which you can add to your app. **Note:** There are various types of restrictions for API keys. You need an API key with restriction for **Android apps** (not a browser-restricted key).

Quick guide to getting a key

Step 1. Get an API key from the Google API Console

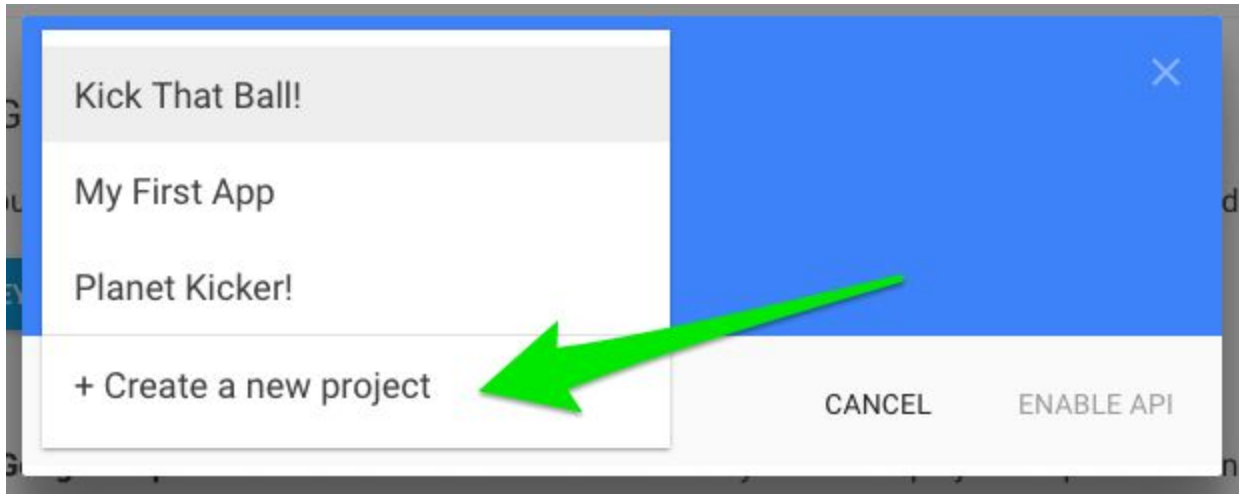
Click the button below, which guides you through the process and activates the Google Maps Android API automatically.

GET A KEY

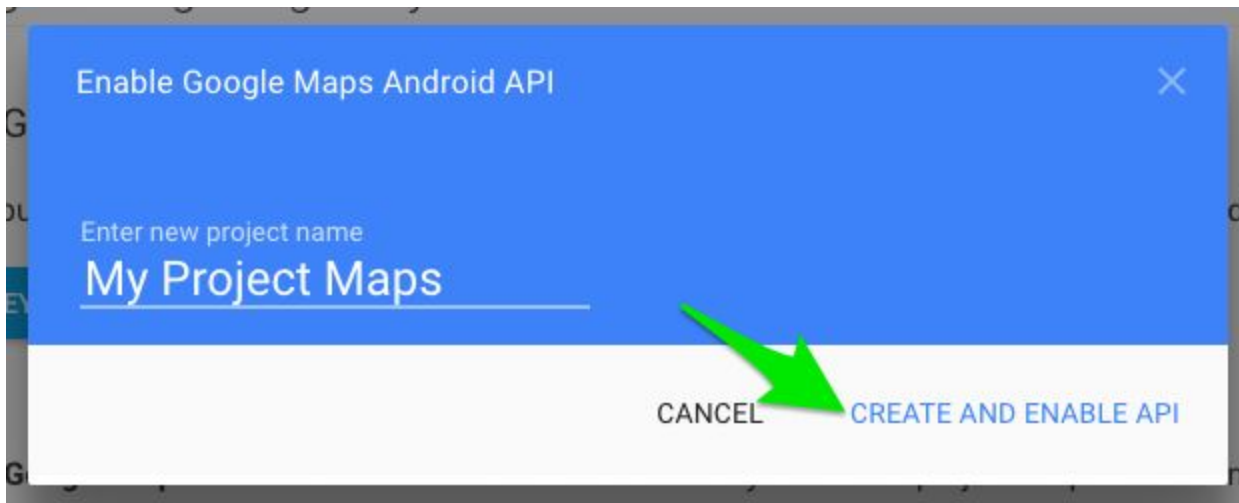
Notes:

- **For Google Maps APIs Premium Plan customers:** When you see the project drop-down menu, you must select the

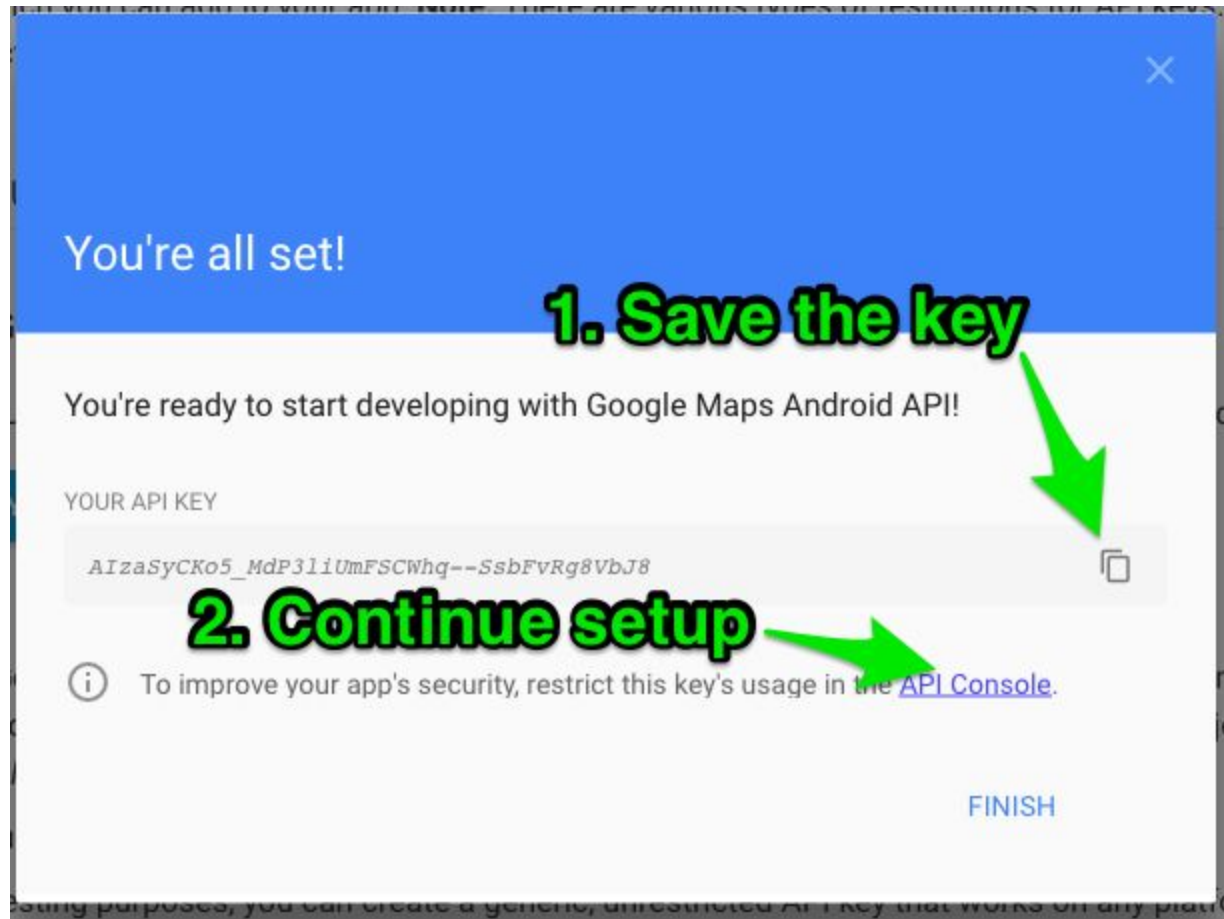
- Click on "Create a new Project" Button



- Click on "CREATE AND ENABLE API" Button



- Now copy your key and save it, you will need it later. After this go to API Console following the link as in image.



- What we need to do now and it is very important that we restrict usage of this key to only our Android application (So other people can't use it if they obtain your key). In Key restriction section choose Android apps and click on + Add package name and fingerprint button.

API Manager Credentials

Dashboard
Library
Credentials

← Regenerate key Delete

API key
This API key can be used in this project and with any API that supports it. To use this key in your application, pass it with the `key=API_KEY` parameter.

Creation date Feb 16, 2017, 7:12:26 PM
Created by leskiv.taras@gmail.com (you)

API key
AIzaSyCKo5_MdP31iUmFSCWhq--SsbFvRg8VbJ8

Name
API key

Key restriction
This key is unrestricted. To prevent unauthorized use and quota theft, restrict your key. Key restriction lets you specify which web sites, IP addresses, or apps can use this key. [Learn more](#)

☐ None
☐ HTTP referrers (web sites)
☐ IP addresses (web servers, cron jobs, etc.)
☒ **Android apps**
☐ iOS apps

Restrict usage to your Android apps (Optional)
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps. Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
$ keytool -list -v -keystore mystore.keystore
```

Note: It may take up to 5 minutes for settings to take effect

Save Cancel

- The form will appear - put Package name from you unity project that you set up in Step 1*.
- To obtain SHA-1 certificate fingerprint run this command in your terminal pointing to your keystore that application is signed with:

keytool -list -v -keystore mystore.keystore

For example in my case it is `keytool -list -v -keystore ~tarasleskiv/.android/debug.keystore` as I am using default debug keystore. (Default option in Unity).

After you run the command you will see the result like in the image below, copy SHA-1 certificate fingerprint into the form in Google Console:

```
Your keystore contains 1 entry

Alias name: androiddebugkey
Creation date: Aug 13, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 448a4568
Valid from: Thu Aug 13 15:28:47 CEST 2015 until: Sat Aug 13 15:28:47 CEST 2045
Certificate fingerprints:
  MD5: 35:32:75:31:60:87:75:35:35:20:05:05:25:31:63:23
  SHA1: 98:82:5B:3A:40:45:02:A6:43:64:16:5D:3E:20:2A:B5:22:4D:01:5E
  SHA256: 3D:27:1B:1D:3C:8E:05:CE:12:07:EC:11:08:1B:0A:9E:47:11:CE:DA:21:04:1B:E:1E:E5:9F:86:08:16:11:80:73:1B:08
Signature algorithm name: SHA256withRSA
Version: 3
```

SHA-1 certificate fingerprint

- After you filled in all the information click Save. Note: It may take up to 5 minutes for settings to take effect

Restrict usage to your Android apps (Optional)

Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps

Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
$ keytool -list -v -keystore mystore.keystore
```

Package name	SHA-1 certificate fingerprint
com.example	12:34:56:78:90:AB:CD:EF:12:34:56:78:90:AB:CD:EF:AA:BB:CC:DD
Add package name and fingerprint	

Note: It may take up to 5 minutes for settings to take effect

[Save](#)

3. Put the API key inside your AndroidManifest.xml inside Unity Project.

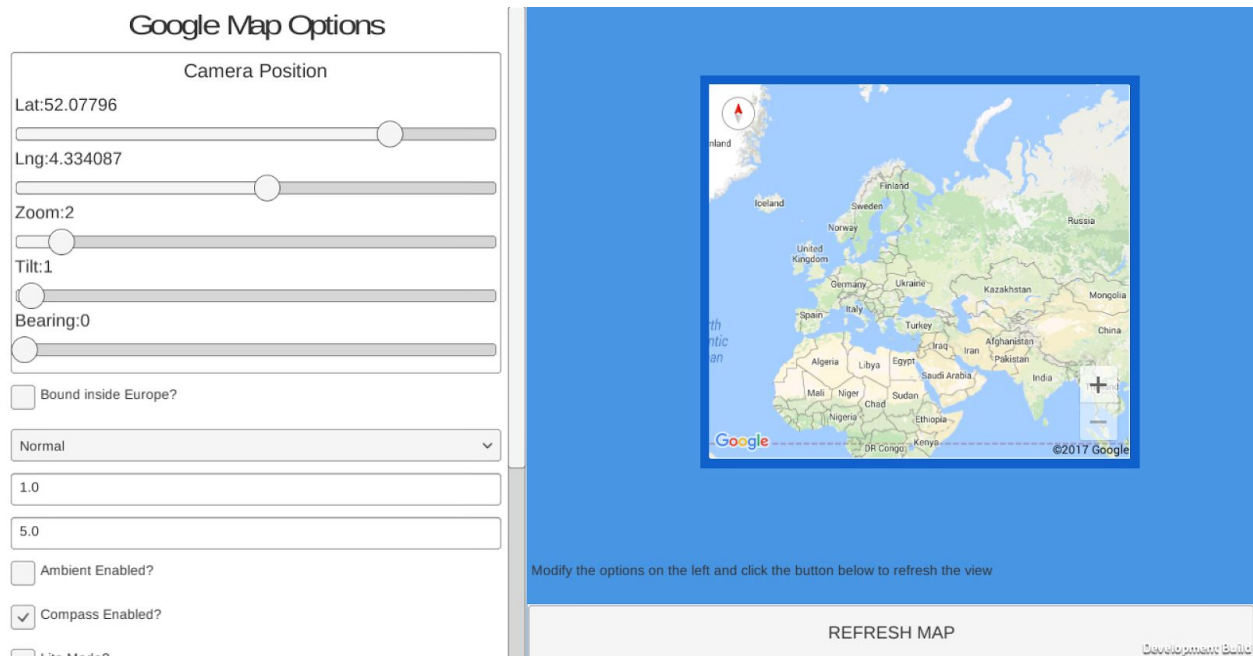
Once you have your key (it starts with "Alza"), put the following meta-data tag INSIDE application tag replacing the value with the API key that you recently retrieved.

```
<meta-data android:name="com.google.android.geo.API_KEY"
android:value="YOUR_GOOGLE_API_KEY_HERE" />
```

4. Run the Demo Scene

- Open Unity Build Settings and switch Platform to Android
- Add **GoogleMapView/Example/ExampleScene.unity** to Scenes in Build
- Connect Android device and run the scene (Device must have Google Play Services installed)

After running the demo you will see the demo scene, now you can play around with settings. Don't forget to click "Refresh" each time you change settings.



Usage Instructions

For reference see the Example folder inside unitypackage. It must provide exhaustive overview of plugin functionality.

Creating, Showing and Dismissing GoogleMapView

To create the google map view simply create an object and pass the created options to the constructor. After this call Show method passing rectangle that represents the position on the screen as a parameter. You can create and show multiple map view instances at the same time.

Minimum code to show the map:

```
var options = new GoogleMapsOptions();  
var map = new GoogleMapView(options);  
var screenPosition = new Rect(10, 10, 300, 300);  
map.Show(screenPosition);
```

Dismissing the map:


```
map.Dismiss();
```

For more detailed usage instructions visit the [Usage Instructions Wiki Page](#).

FAQs and Known Issues

For FAQ and known issues visit the [FAQ and known issues Wiki Page](#).