# Mobile Applications: AdMob

## Create an account with google for AdMob

Start by going to <http://www.google.com/admob/landing/sign-up-003.html?subid=emea-semexp3-r3&gclid=CjwKEAjw1_KwBRDEz_WvncL4jGwSJAAEym0dfwrR_TsZVUM-GYhe71aS74aLSXtPkwo0JaCodSbqkhoCkanw_wcB>

Click on Sign up and follow the steps to create an account.

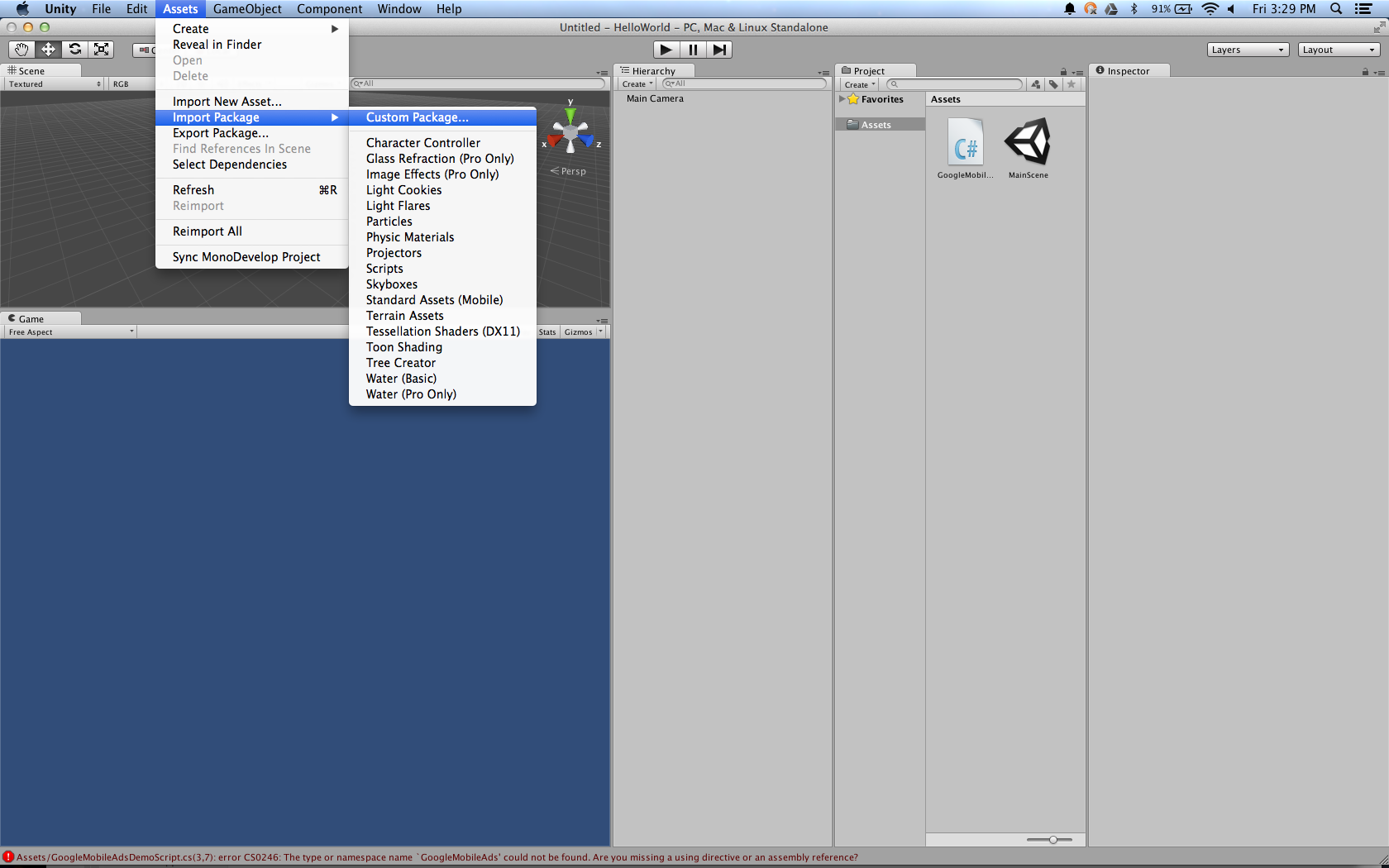
## Setting up AdMob with unity

First you nee to download the plugin for unity. You can get the plugin from <https://github.com/googleads/googleads-mobile-plugins/releases/latest>

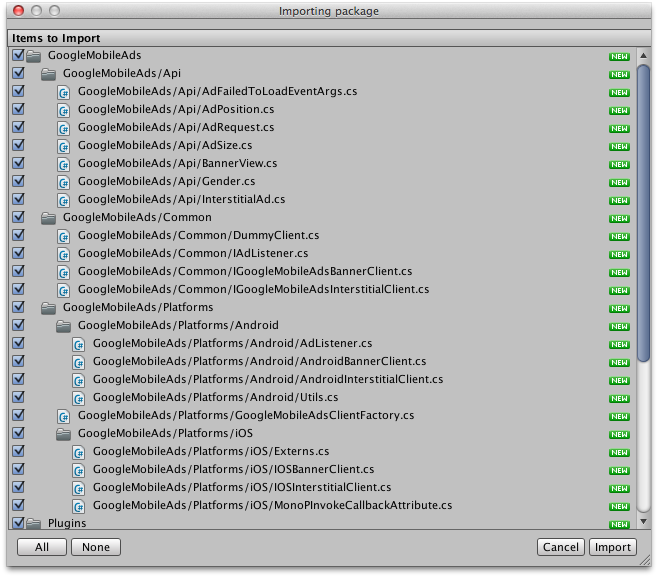
Or

From MyCourse.

Open your project in the Unity editor. Select **Assets > Import Package > Custom Package** and find the GoogleMobileAdsPlugin.unitypackage file you downloaded.



Make sure all of the files are selected and click **Import**.



Next, you need to make a copy of the google play services folder.

This is normally located in <android\_sdk>/extras/google/google\_play\_services/libproject/google-play-services\_lib/

However, dude to restrictions on the PC’s in 609, it doesn’t exist at all.

Instead, you will need to download the Google play services zip file on mycourse.

Download both the GoogleMobileAdsPlugin and the google-play-services lib zip files.

Unzip the folders in to your projects Assets/Plugins/Android folder.

## Coding with AdMob (the fun bit)

### Banner Ads

AdMob has a very simple. Create a new script file and attach it to a game object.

Open the script file

Include the GoogleMobileAds.api

using GoogleMobileAds.Api;

Now create the following function

private void RequestBanner()  
{  
    #if UNITY\_ANDROID  
        string adUnitId = "INSERT\_ANDROID\_BANNER\_AD\_UNIT\_ID\_HERE";  
    #else  
        string adUnitId = "unexpected\_platform";  
    #endif  
  
    // Create a 320x50 banner at the top of the screen.  
    BannerView bannerView = new BannerView(adUnitId, AdSize.Banner, AdPosition.Top);  
    // Create an empty ad request.  
    AdRequest request = new AdRequest.Builder()  
 .AddTestDevice(AdRequest.TestDeviceSimulator)       // Simulator.  
   .AddTestDevice("INSERT DEVICE ID HERE")  // test device.  
 .Build();  
    // Load the banner with the request.  
    bannerView.LoadAd(request);  
}

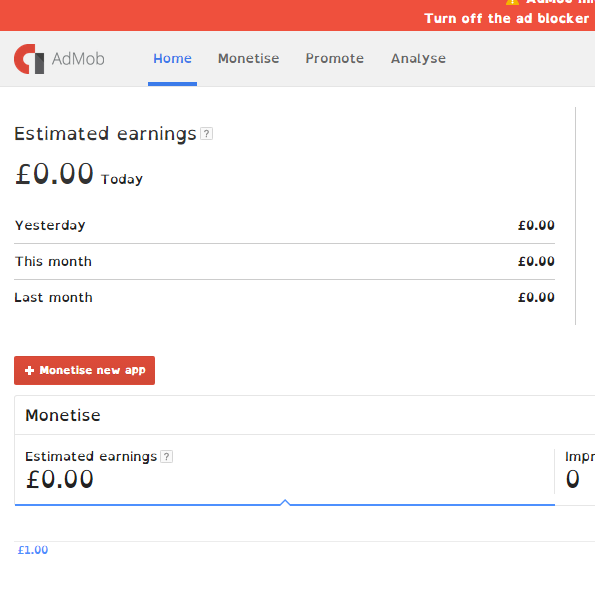
This code will automatically select the correct ad ID, for the current running platform.

You will need to change the “INSERT DEVICE ID HERE” to the hardware ID of the device.  
to do this you need to go to the following section of this document, Setting up test ads and follow the steps to find the device ID.

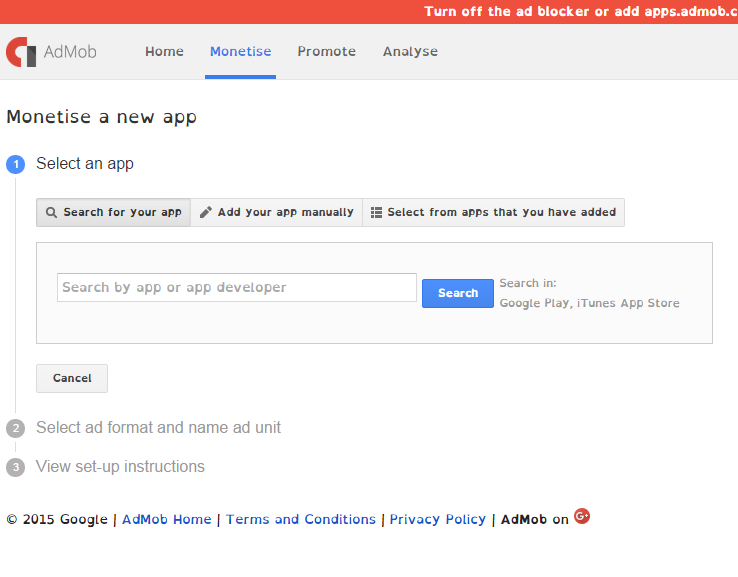
At the moment it’s “INSERT\_ANDROID\_BANNER\_AD\_UNIT\_ID\_HERE”

Delete this and go back to the admob page.

Click on the “monetise new app” button



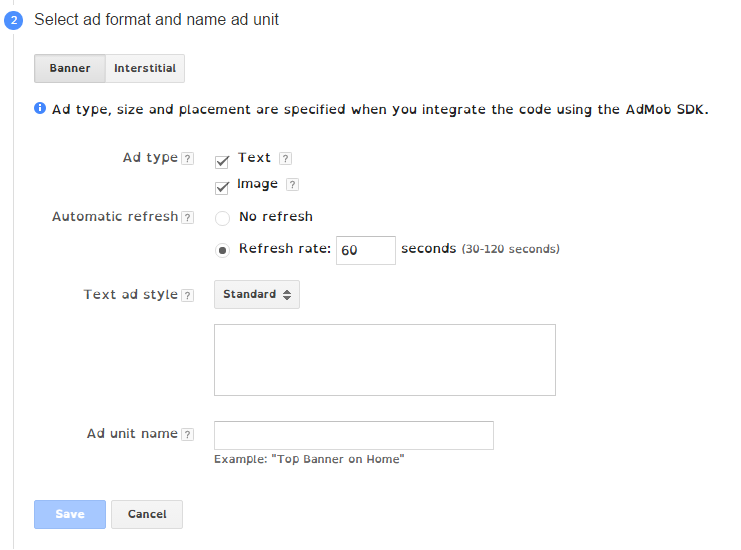
On the next screen click on “add app manually”



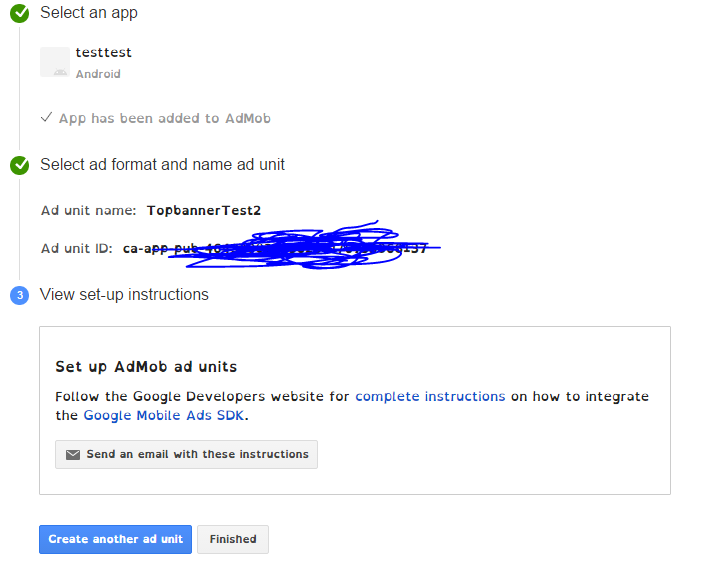
Enter a name for your app, select android as the platform and click “add App”

Now select “Banner”

And type a name for the new banner, make it something descriptive.



And click save



Copy the ad unit ID code and past it in to the adUnitId string in your code.

Now make sure you call the RequestBanner function from the start function, save run you code (for android!).

You should see a banner add appear in the top of the game screen

### Interstitial ads

The following code requests an interstitial ad

private void RequestInterstitial()  
{  
    #if UNITY\_ANDROID  
        string adUnitId = "INSERT\_ANDROID\_INTERSTITIAL\_AD\_UNIT\_ID\_HERE";  
    #else  
        string adUnitId = "unexpected\_platform";  
    #endif  
  
    // Initialize an InterstitialAd.  
    InterstitialAd interstitial = new InterstitialAd(adUnitId);  
    // Create an empty ad request.  
    AdRequest request = new AdRequest.Builder().Build();  
    // Load the interstitial with the request.  
    interstitial.LoadAd(request);  
}

Again, you need to replace the “INSERT\_ANDROID\_INTERSTITIAL\_AD\_UNIT\_ID\_HERE” string with a new Ad unit ID.

Generate another Ad unit ID just as you did for banner ads but this time, select “Interstitial” instead of “banner” during step 2 of the Ad unit ID creation.

Insert the new Ad Unit ID in to the string.

Unlike banner ads, interstitial ads need to manually show the interstitial ad, using something similar to the following code

if (interstitial.IsLoaded()) {  
   interstitial.Show();  
 }

Place this code in your update function and it will only run when the ad has downloaded and is ready to be shown,

## Setting up test ads

To avoid generating false impressions, admods recommends that all developers request test ads from there service. To do this, you need to alter you AdRequestBuilder to look similar to the following

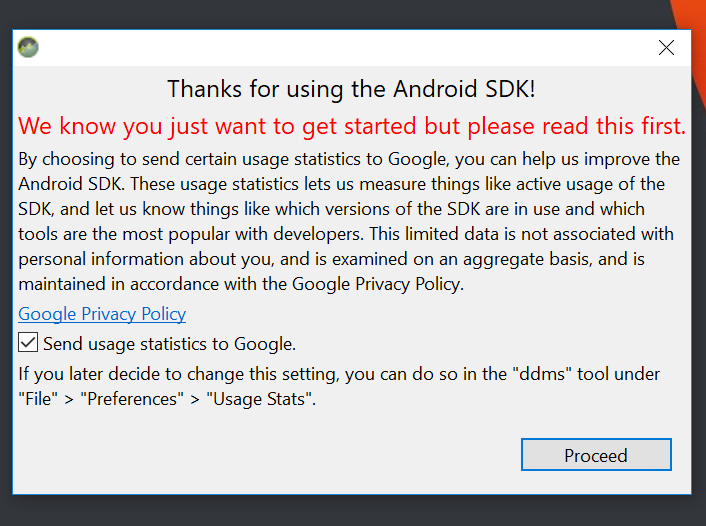
AdRequest request = new AdRequest.Builder()  
    .AddTestDevice(AdRequest.TestDeviceSimulator)       // Simulator.  
    .AddTestDevice("2077ef9a63d2b398840261c8221a0c9b")  // My test device.  
    .Build();

This code will request ads for testing purposes.

Notice the 4th line, with the long string of alphanumeric text.

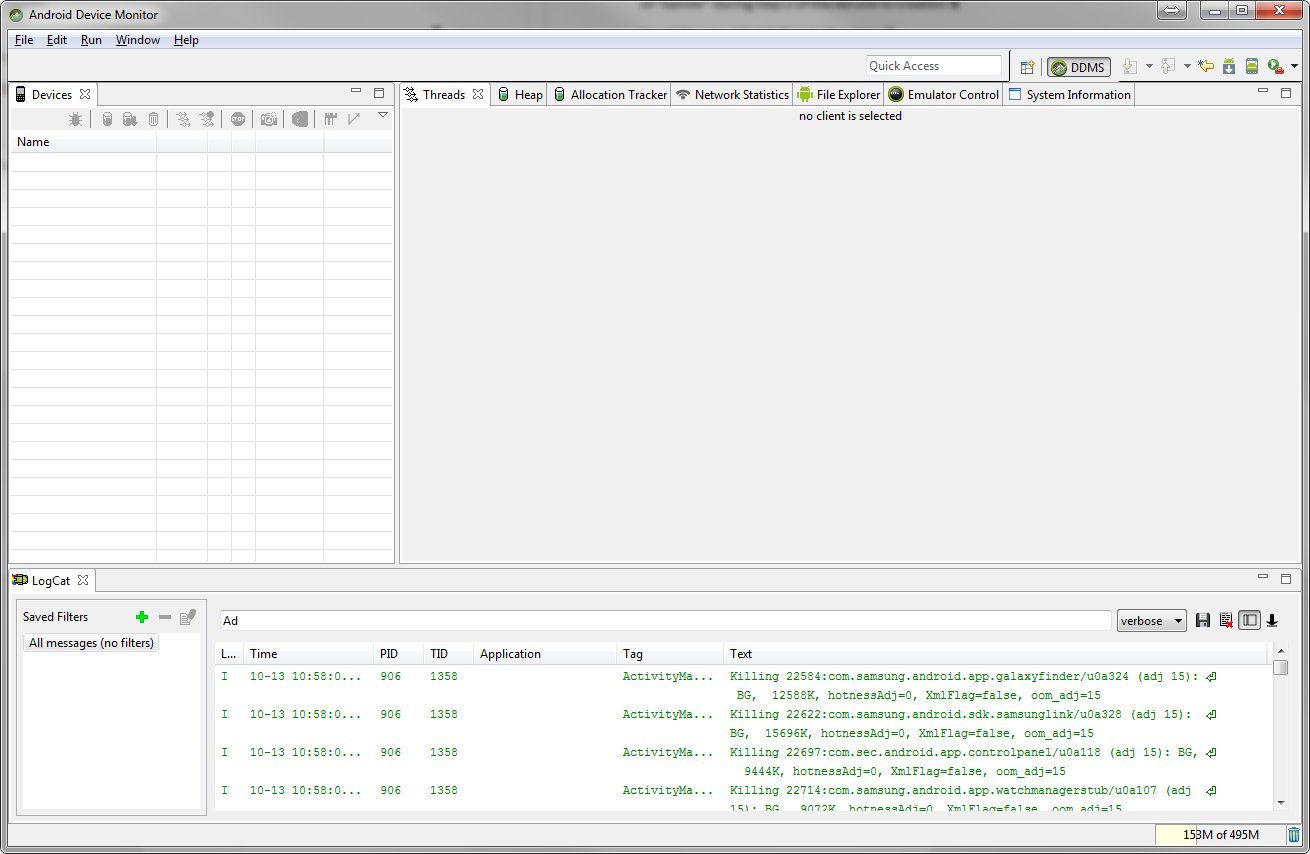
You need to replace this with the encrypted device ID for you mobile device.

To find you device ID, navigate to C:\Program Files (x86)\Android\android-sdk\tools\lib\monitor-x86\_64 and run the monitor.exe file

A console window will briefly open and tell you that registry editing is not allowed, before closing. It will seem like nothing has happened, however, press alt + tab you will see a window like this. 

Select this window and press “Proceed”

You will see the following window



Make sure you have selected you attached device from the list on the left and in the small text box in the centre of the Logcat tab, at the bottom of the screen, type “Ad”

Now, in Unity, run your project and watch the log cat window for the following message

Ads(2132): To get test ads on this device, call adRequest.addTestDevice("D9XXXXXXXXXXXXXXXXXXXXXXXXXXXXX");.

Copy the long sting of alpha numeric text and past it over the device ID number in you code. This is your devices encrypted ID.

Incidentally, if you want to us Debug.Log statements while testing you apps on a device, you can get view there output using this same method. Just remove the “Ad” text from the text box of the log cat tab and wait until you see you message appear in the log.

## Ad Events

Admobs allows you to access to register for several events that its objects my fire.

These events are

AdLoaded

AdFailedToLoad

AdOpened

AdClosed

AdClosing

AdLeftAplication (fires when the user leaves the application via an ad).

The following is a simple example of how to hock in to the events

private void RequestBanner()  
{  
    BannerView bannerView = new BannerView(adUnitId, AdSize.Banner, AdPosition.Top);  
    // Called when an ad request has successfully loaded.  
    bannerView.AdLoaded += HandleAdLoaded;  
}  
  
public void HandleAdLoaded(object sender, EventArgs args)  
{  
    print("HandleAdLoaded event received.");  
    // Handle the ad loaded event.  
}

## Banner visibility

It is possible to show and hide banners via code, using

bannerView.Hide();

And

bannerView.Show();

By default, BannerViews are visible from the get go.

## Clean up

Once you have finished with the Banner and interstitial ads, you should call Destroy() on the interstitialAd and BannerView objects. Failure to do so will result in memory leaks.



## Task

Change the position of the banner ad

Change the size of the banner ad