Windows Unity Ad Network Plugin

A simple, flexible, easy to use plugin for Unity3D that allows Ads to be displayed in Windows Phone & Windows Store Apps

Setting up the plugin:

Unity

Import the package into your Unity project.

Add either the **AdMobAd** or **PubCenterAd** prefab into your scene.

Once you have added them into your scene you must enter your correct credentials (AdUnitID, ApplicationID etc), the width/height of the ad and in the case of AdMob the format (Banner, or SmartBanner) See AdMob documentation

You can set the position of your ad with the *positionX* and *positionY* variables. These values are screen percentages, that are in the coordinate system of a WP8/Windows Store app. This means the values are from -50 to 50, zero being center.

If the variables are not showing up in the Inspector of Unity, make sure the project is set to the right platform. You can set this by selecting File->Build Settings (Ctrl-Shift-B)

Testing

For PubCenter Ads you can leave the default values provided and you will get test ads. Currently using actual values will not work in the emulator, but they will work on a device. Windows Store test values, Windows Phone test values for PubCenter Ads.

For AdMob, you will need to change the values. If you check the "Test Ad Mob" check box it will do an *adRequest* (AdMob explanation). This is important to use during development to avoid generating false impressions. **Remember don't leave this selected when you submit the game to the store.**

AdFiller

The AdFiller is for when no ad is available. An *adFiller* consists of an image and an url, allowing you to direct traffic to your site or another app/game. Use this responsibly, and comply with all EULAs

Once everything is set up, go head and hit build.

For debugging purposes errors will be printed out, you can turn this off by un-checking the printDebug.

Windows Phone Apps

First off, in Visual Studio, open your **WMAppManifest.xml** and add these capabilities

Required Capabilities (AdMob)

- ID_CAP_NETWORKING - Access to network services is required when requesting ads.
- ID_CAP_WEBBROWSERCOMPONENT - Required since the AdView is a web browser.
- ID_CAP_MEDIALIB_PLAYBACK - Provides access for currently playing media items.
- ID_CAP_MEDIALIB_AUDIO - Provides read access to audio items in media library.

Required Capabilites (PubCenter)

- ID_CAP_IDENTITY_USER
- ID_CAP_MEDIALIB_PHOTO
- ID_CAP_NETWORKING

- ID_CAP_PHONEDIALER
- ID_CAP_WEBBROWSERCOMPONENT

Next, go to your references, we will need to make some changes

Remove both of the **Microsoft.Advertising** dlls.

Bring in the **Windows Phone Ad SDK** for XAML. Add a reference to the **WindowsAdPlugin** in the *assets/plugins/wp8* folder

Now open the **MainPage.xaml.cs** and lets add some code to hook up the Dispatcher class to handle invokes on the App or UI thread

First off add these two functions, that tie into the Unity App thread and the UI thread of the solution

```
public void InvokeOnAppThread(Action callback)
{
    UnityApp.BeginInvoke(() => callback());
}

public void InvokeOnUIThread(Action callback)
{
    Dispatcher.BeginInvoke(() => callback());
}
```

Then add these lines of code, within the UnityLoaded function to wire it all up

```
WindowsHelperPlugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
WindowsHelperPlugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
```

Almost done, next we need to pass in a grid for the ad to be added to. For this we use the DrawingSurfaceBackground

Example:

```
WindowsHelperPlugin.Helper.Instance.SetGrid(DrawingSurfaceBackground);
```

Now we can run the game and see the ads appear

Windows Store Apps

Note: only pubCenter ads are currently working in Windows Store games

In Visual Studio, open **Package.appmanifest** and the following required capabilities

- Internet(client)

Now lets turn our attention to our references. First delete the **Microsoft.Advertising.WinRT.UI** and **Microsoft.Advertising.WinRT** references. Then add our advertising reference, go to **Windows->Extensions->Microsoft Advertising SDK for Windows 8.1(XAML)**

Next we will move on to the code. Go into **App.xaml.cs** and paste this:

```
appCallbacks.Initialized += appCallbacks_Initialized;
```

Now in the constructor, then paste this wherever inside that script

```
void appCallbacks_Initialized()
{
    Windows_Ad_Plugin.Dispatcher.InvokeOnAppThread = InvokeOnAppThread;
    Windows_Ad_Plugin.Dispatcher.InvokeOnUIThread = InvokeOnUIThread;
}
```

Next lets add the hookups so we can invoke on the UI and App thread

```
public void InvokeOnAppThread(Action callback)
{
```

```
        appCallbacks.InvokeOnAppThread(() => callback(), false);
    }
    public void InvokeOnUIThread(Action callback)
    {
        appCallbacks.InvokeOnUIThread(() => callback(), false);
    }
```

Finally lets go to your **MainPage.xaml.cs** and set the Grid for the Plugin to add the ad to

```
    Windows_Ad_Plugin.Helper.Instance.SetGrid(DXSwapChainBackgroundPanel);
```

That's it! If we run the game the ads will now appear