Android Native

Introduction

Android Native is designed to maximize the functionality of Android devices within Unity3D. It does not require Unity Pro, or Android Pro. You do not need to be an expert coder in any way, but the basics will help. Our goal of this project is to minimize coding by simply calling the "Helper" files from your scripts. Once your Scene includes one of these Helper files, you can simply call a function within that file. This should be achievable with just a few lines of code. We will show you a very basic example below, but we suggest you study the included code.
An example scene is also included with this project. You can simply build it, and run it. You can also download it from the Market here. If you have any questions or wish to requests additional functionality to be added to the next update, please email us at DevfoMobile@gmail.com

A few items to mention:

- Since this is a native Android plugin, it will not run inside Unity. You will need to build the scenes and deploy them to an Android emulator or device. But don't worry if you try it with our sample scenes, the calls will not be made.
- Always call AndroidJNI.AttachCurrentThread() before you make a call to the Android Native library. It may not always be needed, but you want to make sure it's attached.

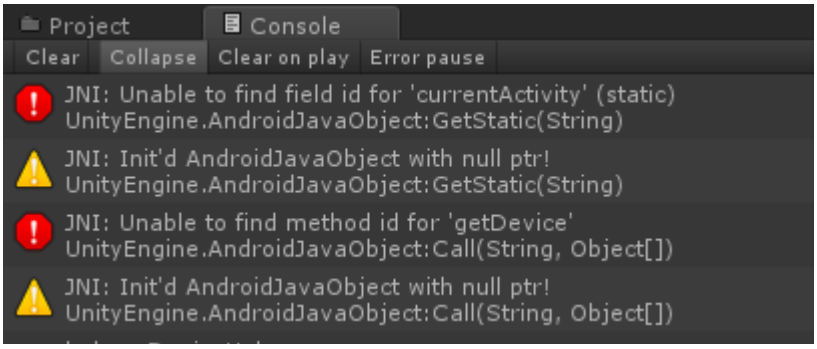
First Step – Creating a very simple scene

1. Import the AndroidNative package. If you are reading this, you probably already have.
2. Create a new empty scene.
3. Create a new game object (menu -> game object -> create empty).
4. Create a new C# script (let's call it TestScript.cs). And attach it to your newly created game object. Note that you cannot attach one of the "Helper" scripts directly to the object. This is because they are not extending the MonoBehavior. However you may wish to write your own scripts to best fit your project.
5. Now open your TestScript.cs file and add a simple test function:

```
public class TestScript : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }

    void myTestFunction () {
        DeviceHelper helperScript = new DeviceHelper();
        string deviceId = helperScript.GetDeviceId();
    }
}
```

6. Now just create a simple button that calls your new function. That's it….However your project will most likely not be this simple. So we suggest that you take a look at the DDevice.cs script. These will show you the proper way to instantiate your objects on the Awake() function.

Remember, if you try to run this within Unity, you will an error like this:



What's included

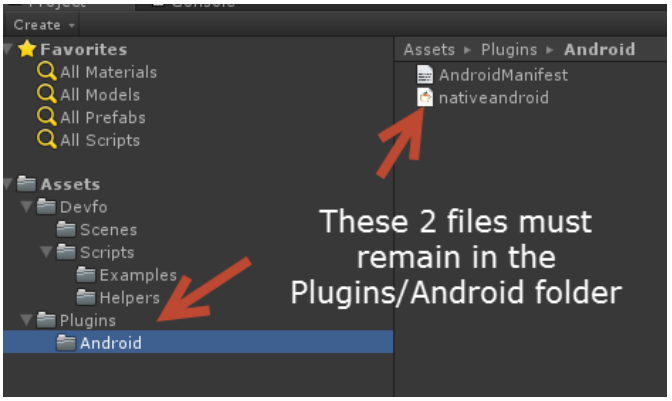
The following Helper files are included. Although they are not necessary and you may find yourself writing your own, they are a good place to start. Below each helper file, the available functions. . All functions are fully commented it their corresponding Helper files. Please take a look at the Helper files for further understanding.

- ContactsHelper.cs
  - GetContactDetailsFromPhone(string phoneNumber)
  - GetContactDetailsFromEmail(string emailAddress)
- DeviceHelper.cs
  - GetDeviceId ()
  - GetVersionName()
  - GetVersionCode()
  - HasConnection()
  - GetDevicePhoneNumber()
  - GetVmHeapStats()
  - GetCountryCode()
- DialogHelper.cs
  - ShowNativeDialog(string message)
  - howNativeDialogOK(string message)
  - ShowNativeDialog(string message, string[] button1)
  - ShowNativeDialog(string message, string[] button1, string[] button2)
  - ShowNativeDialog (string message, string[] button1, string[] button2, string[] button3)
- PackageManagerHelper.cs
  - IsSafeMode()
  - HasPermission (string permissionName, string packageName)
  - GetInstalledApplications (bool getSysPackages)
  - RemoveAppIcon ()
  - StartApplication (string application)
  - GetPackageInfo (string packageName)
  - addPermission (int[] permissions, bool async)
  - RemovePermission (string permission)
- ProgressDialogHelper.cs

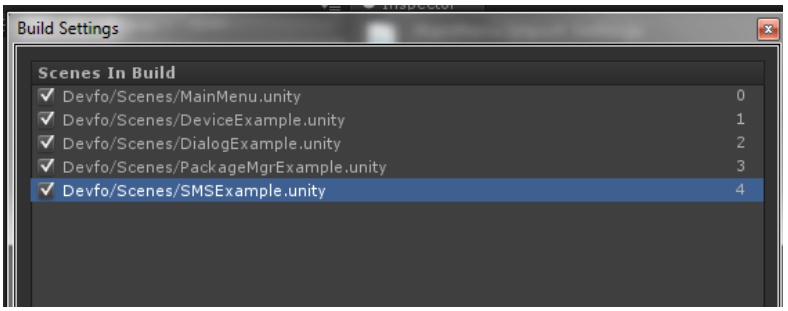
- ShowSpinnerDialog(string title, string message)
- ShowSpinnerDialog(string title, string message, bool cancelable)
- ShowSpinnerDialog(string title, string message, bool cancelable, string[] cancelCallback)
- ShowProgressDialog(string title, string message, bool cancelable, string[] cancelCallback, int maxBarValue, bool autoClose)
- UpdateProgressDialog(int progress)
- DismissProgressDialog()

- SMSHelper.cs
  - LaunchSMSActivity ()
  - LaunchSMSActivity (string number, string msg)
  - LaunchSMSActivity (string number)
- ToastHelper.cs
  - showToastShort(string message)
  - showToastLong(string message)

Inside the Plugins/Android folder, you will find 2 files. One is the NativeAndroid.jar file, and the other is the Android Manifest. You shouldn't need to modify the Manifest file UNLESS you currently have a Manifest file. If so, then you may need to merge both manifest files together. If you need help with this, search the posts on the Unity forums.



Well, we hope this will help you get started. Just take a look at the included sample scenes. You can run each scene (ie: DialogExample.scene) on its own. But if you want to build all scenes with the MainMenu.scene, make sure you set up your build settings correctly, like this:



Changelog

Version 1.1

- Initial Release