# MOGA Unity Plugin Documentation

**Summary**

The nature of this plugin allows us to create a customised version of Unity's built in input manager to explicitly map our MOGA Controller to match the conventional unity functionality.

The Contents of this documentation provide detailed instructions on how to setup the MOGA Controller with the Unity3D Game Engine for Android and Windows Phone 8.

A YouTube tutorial for this tutorial can be found here.

**Version:** 0.5
**Date:** 02/12/13

## Contents

History 4
**1. Windows Phone 8 4**
1.1 WP8 Plugin Essential Files 5
1.2 Creating the MOGA Controller Manager 6
1.3 Attaching the MOGA Input Script 7
1.4 Mapping the Controls (optional) 8
1.5 Final Steps and One Click Build Process 9
1.6 Manual Visual Studio Build (optional) 10
1.6.1 Reference the Library 10
1.6.2 Add the Unity Plugin to Visual Studio 11
1.6.3 Update MainPage.xaml 12
1.6.4 Update App.xaml.cs 13
1.6.5 Update WPAppManifest.xml 14
1.7 Tips 15
1.7.1 Polling Connection Status 15
1.7.2 Identifying Controller Model 15
1.7.3 Enable GPS 15
**2. Android 16**
2.1 Android Plugin Essential Files 17

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

2.2 Creating the MOGA Controller Manager 18
2.3 Attaching the MOGA Input Script 19
2.4 Mapping the Controllers (optional) 20
2.5 Managing the Controls 21
2.6 Tips 22
**3. Integration Tutorial 23**
3.1 Adding MOGA Android/WP8 Support to AngryBots 23
3.1.1 Importing the MOGA Plugin 23
3.1.2 Change Project Build to WP8 24
3.1.3 Open the MOGA Controller Setup Window 24
3.1.4 Attach the MOGA_Input Script to the Player GameObject 25
3.1.5 Edit Scripts > Movement > PlayerMoveController.js 26
3.1.6 Edit Scripts > Movement > PlayerMoveController.js (continued) 27
3.1.7 Edit Scripts > Modules > TriggerOnMouseOrJoystick.js 28
3.1.8 Build and Run 29

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## History

| Version | Author | Date | Description | Sections Affected |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| **v0.5** | MD | 02/12/2013 | Support for Unity3. | |
| **v0.2** | MD | 25/10/2013 | Added JavaScript to Android, YouTube Link | 2.5 |
| **v0.1** | MD | 17/10/2013 | First Draft. | All |

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## Windows Phone 8

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.1 WP8 Plugin Essential Files

Implementation of the Unity Controller requires the following files;

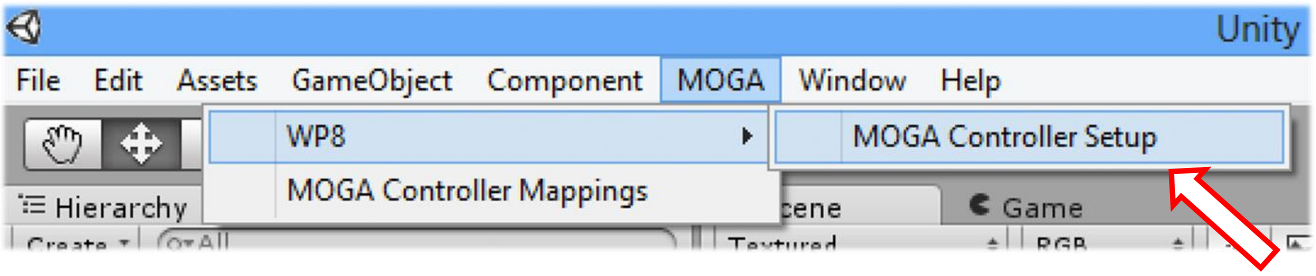
| | | |
|---|---|---|
| Standard Assets > | Scripts > | Moga_Controller.cs |
| | | Moga_ControllerManager.cs |
| | | Moga_Input.cs |
| Editor > | | Moga_Mapping.cs |
| | | Moga_Setup.cs |
| | | PostProcessBuildPlayer_Moga.cs |
| | | Texture_EditorHeader.psd *optional |
| Plugins > | winphone8 | *.* |

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

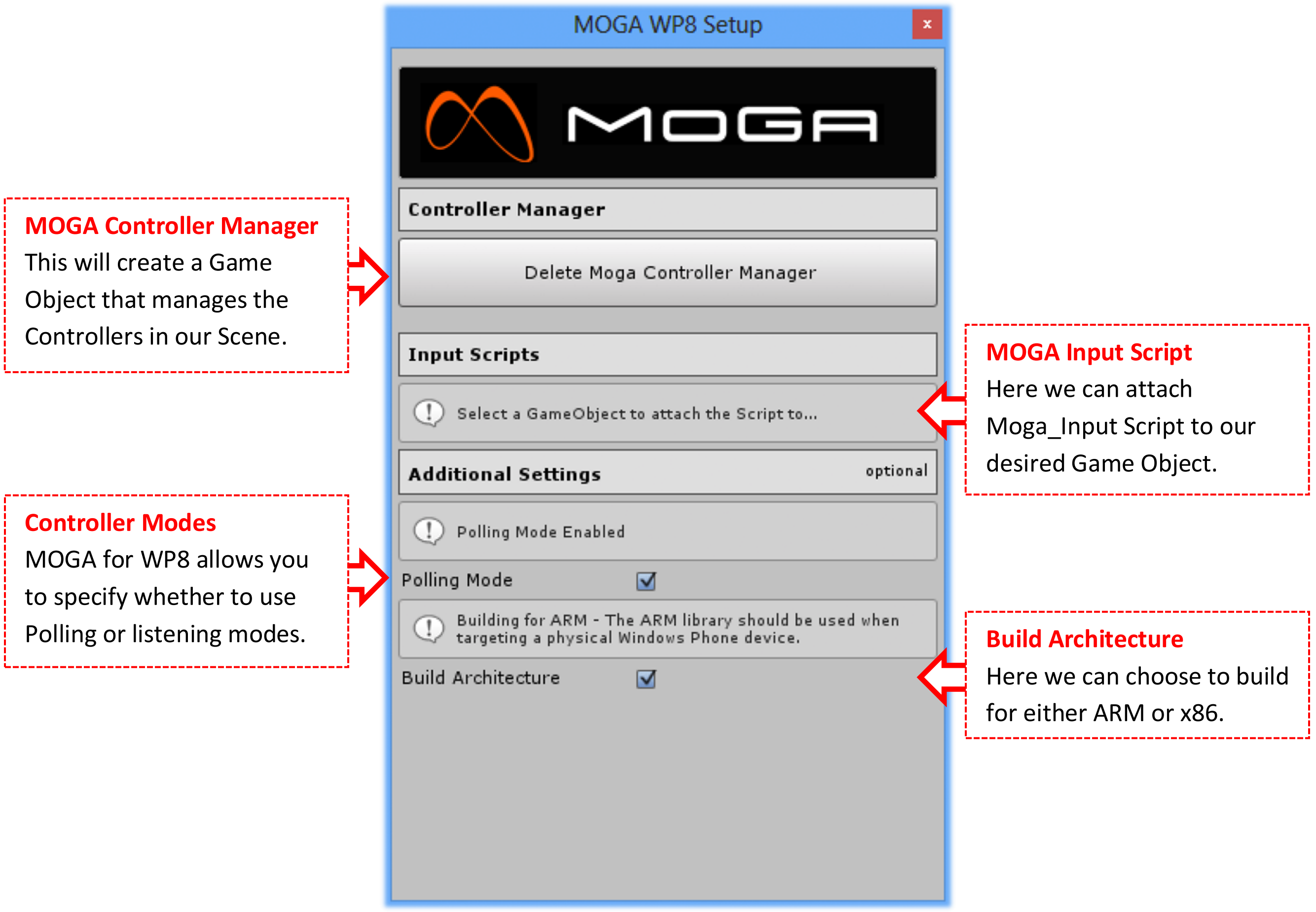
## 1.2 Creating the MOGA Controller Manager

The MOGA Controller Manager will allow us to manage all the Controllers in our application. If you don't want to manually configure your controller Key Mappings then you can skip this process.

Click the **MOGA WP8** Menu Item on the Unity Menu Bar and Select **MOGA Controller Setup** from the dropdown menu.



This will bring up the **MOGA Setup Editor Window** which will allow us to manage our Controller Configuration,


MOGA WP8 Setup
MOGA
Controller Manager
Delete Moga Controller Manager
Input Scripts
Select a GameObject to attach the Script to...
Additional Settings
optional
Polling Mode Enabled
Polling Mode
Building for ARM - The ARM library should be used when targeting a physical Windows Phone device.
Build Architecture

**MOGA Controller Manager**
This will create a Game Object that manages the Controllers in our Scene.

**Controller Modes**
MOGA for WP8 allows you to specify whether to use Polling or listening modes.

**MOGA Input Script**
Here we can attach Moga_Input Script to our desired Game Object.

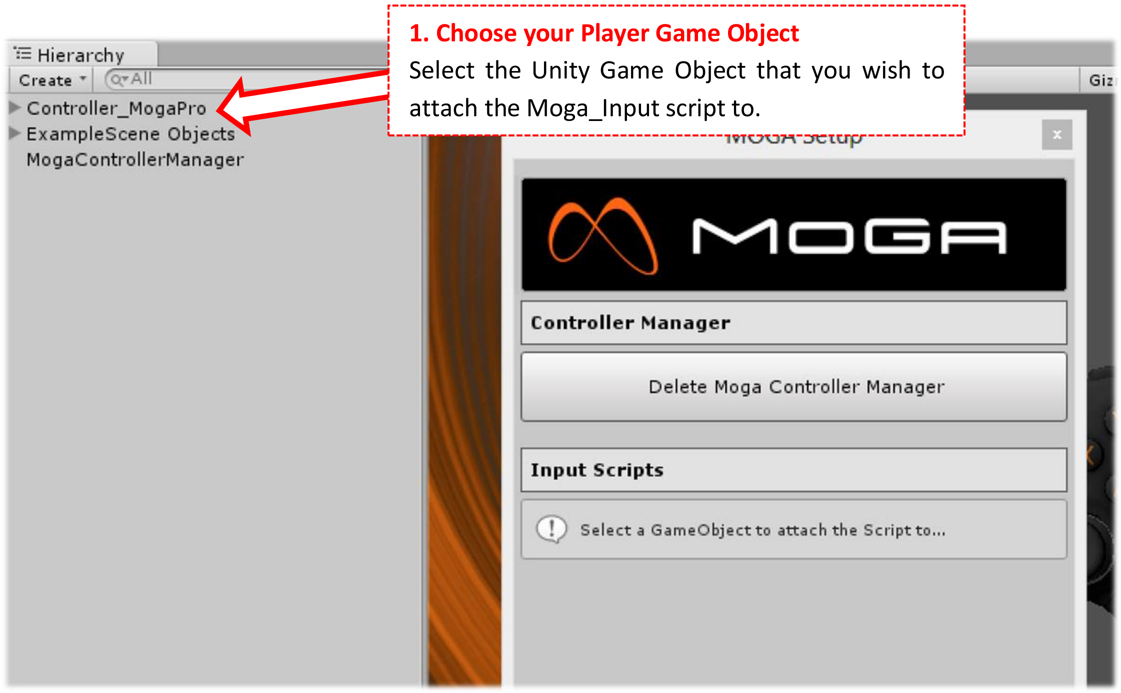
**Build Architecture**
Here we can choose to build for either ARM or x86.

Click **Create MOGA Controller Manager**.
You will notice a **MogaControllerManage**r game object has been added to your project, with an attached script.

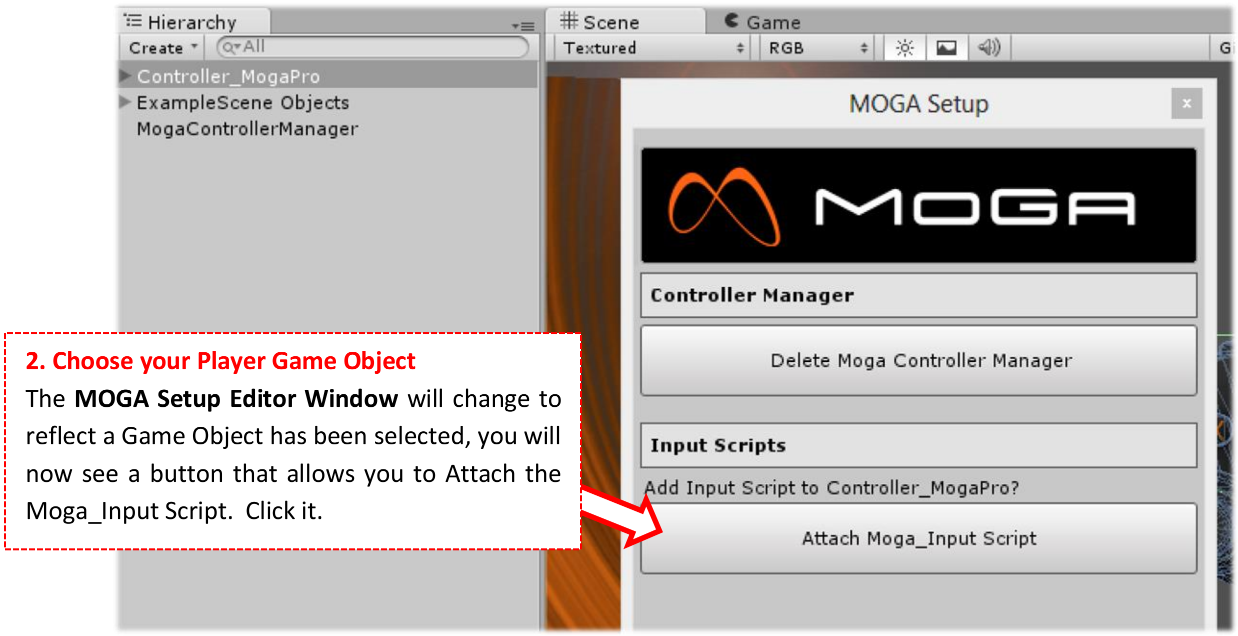
©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.3 Attaching the MOGA Input Script

Within the **Project Hierarchy**, select the Game Object that contains the Script for your controls.



**1. Choose your Player Game Object**
Select the Unity Game Object that you wish to attach the Moga_Input script to.

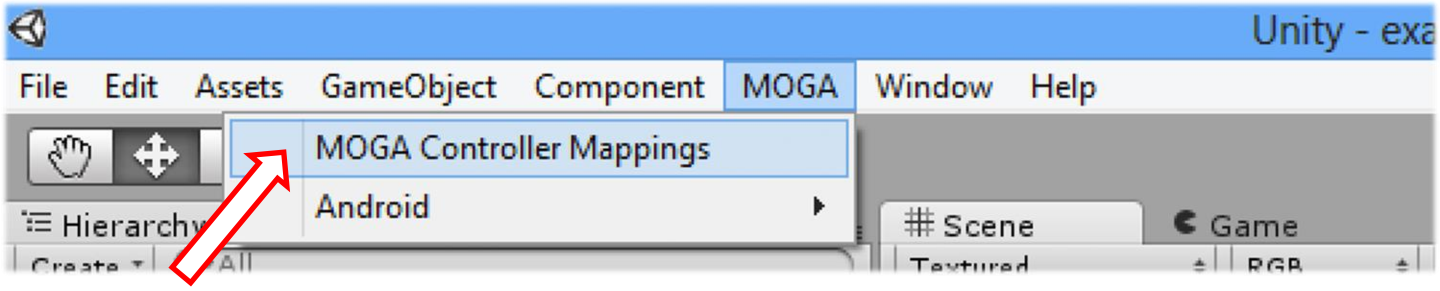


**2. Choose your Player Game Object**
The **MOGA Setup Editor Window** will change to reflect a Game Object has been selected, you will now see a button that allows you to Attach the Moga_Input Script. Click it.

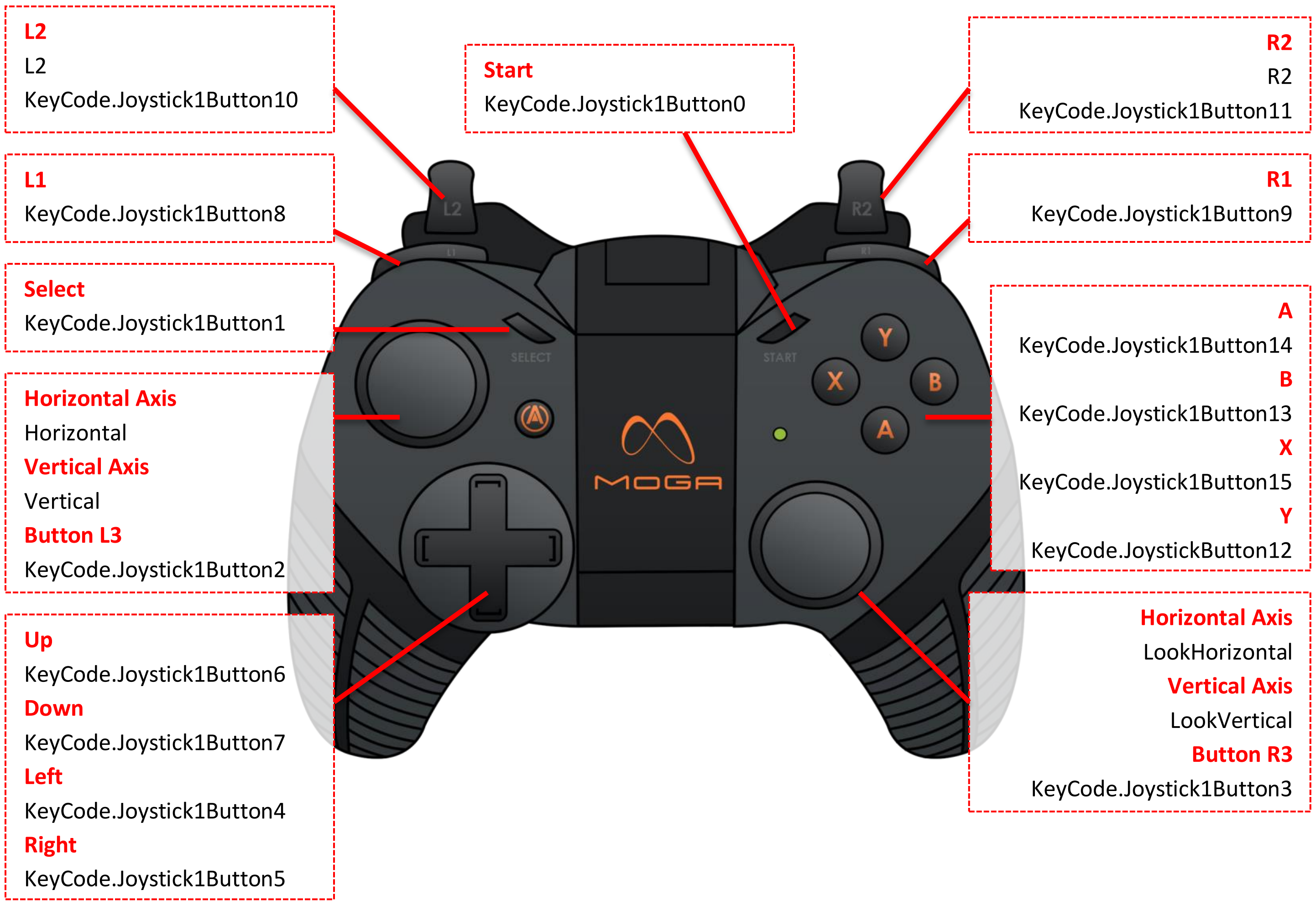
©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.4 Mapping the Controls (optional)

The final part of the MOGA Setup Editor Window allows you to map your Controller buttons and axis. You can omit this entirely and the controls will revert to the default mappings.



If you choose not to create the MOGA Controller Manager, then keys will be assigned by default,



©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.5 Final Steps and the One Click Build Process.

Open the Script that manages the Controls for your Unity Project. This should be attached to the same Game Object that you attached the Moga_Input Script to in Step 1.3.

At the very top of the Script just below your namespaces, add the following line;

```
 #if  UNITY_WP8
using Input = Moga_Input;
#endif
```

Next in your Start method you will need to register your Controller to create the Key Mappings, as show below;

```
void Start ()
{
    // Register MOGA Controller
    #if  UNITY_WP8
    Input.RegisterMogaController();
    #endif
}
```

The MOGA Plugin features a one-click automated build process whereby your Visual Studio project will be automatically updated with the necessary code, libraries and references.

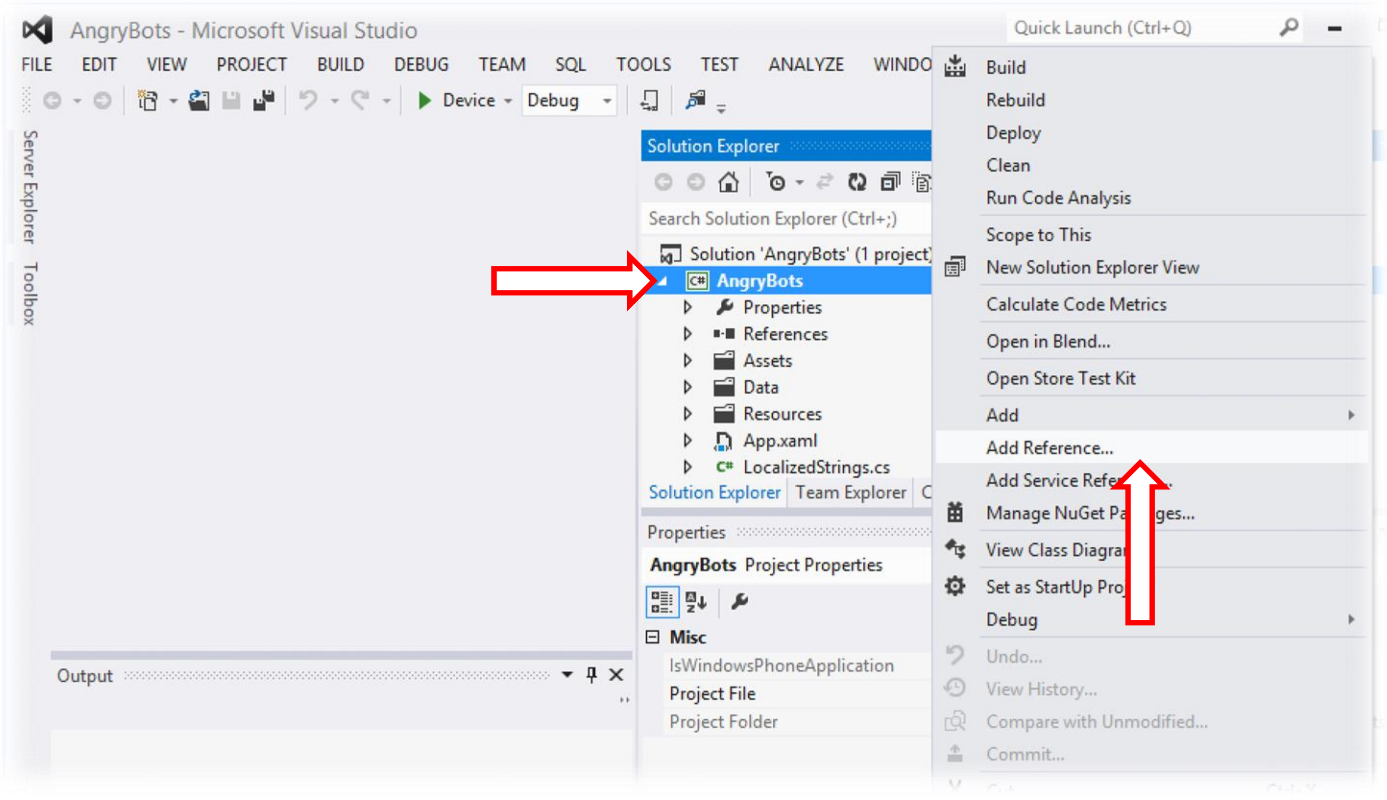
Tip - If you would like to manually build your Visual Studio project you can delete the **PostProcessBuildPlayer_Moga.cs** script from your Editor folder and follow the steps in 1.6.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.6 Manual Visual Studio Build (optional)

### 1.6.1 Reference the Library

If you would like to manually build your Visual Studio project, delete the **PostProcessBuildPlayer_Moga.cs** script from the Editor folder.



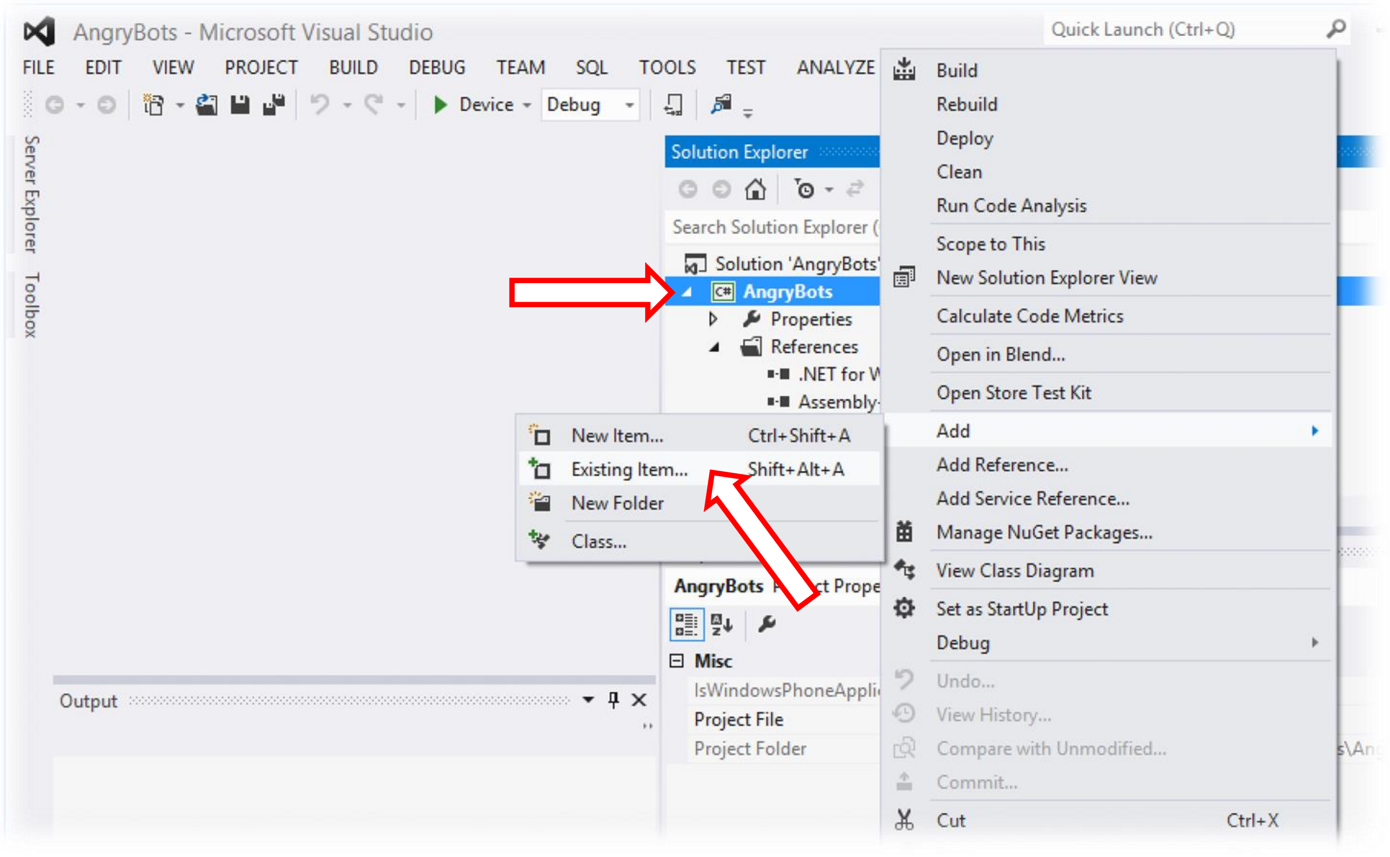
Using **Visual Studio**, open the Visual Studio Project that you built with Unity.

First we need to reference the library**.**

- **Right click** on the **Project name** in the **Solution Explorer Pane**
- Select **Add Reference** from the popup menu.
- Browse to the file **Moga.Windows.Phone.winmd** and select **Add**.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

### 1.6.2 Add the Unity Plugin to Visual Studio

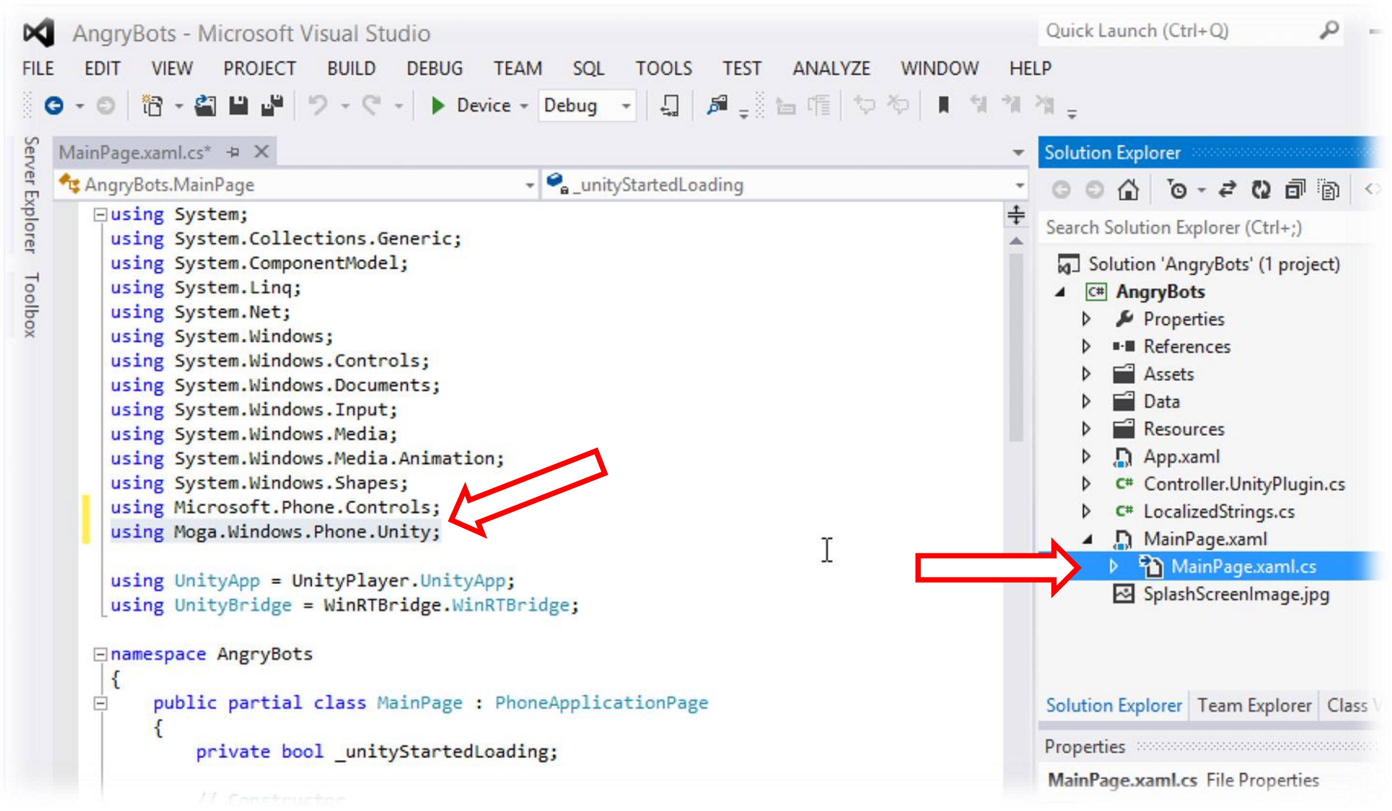


As with the previous step,

- **Right click** on the **Project name** in the **Solution Explorer Pane.**
- Select **Add > Existing Item**.
- Browse to the file **Controller.UnityPlugin.cs** and select **Add**.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.6.3 Update MainPage.xaml


AngryBots - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Device Debug
MainPage.xaml.cs*
AngryBots.MainPage
_unityStartedLoading
Server Explorer Toolbox
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AngryBots' (1 project)
AngryBots
Properties
References
Assets
Data
Resources
App.xaml
Controller.UnityPlugin.cs
LocalizedStrings.cs
MainPage.xaml
MainPage.xaml.cs
SplashScreenImage.jpg
Solution Explorer Team Explorer Class
Properties
MainPage.xaml.cs File Properties

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Moga.Windows.Phone.Unity;

using UnityApp = UnityPlayer.UnityApp;
using UnityBridge = WinRTBridge.WinRTBridge;

namespace AngryBots
{
    public partial class MainPage : PhoneApplicationPage
    {
        private bool _unityStartedLoading;
```

Open the file **MainPage.xaml.cs** from the solution explorer; you will need to **click the dropdown arrow** on the left of **MainPage.xaml** to see this file. The file will open in the Main pane, add the following;

Find the below,

```
using Microsoft.Phone.Controls;
```

And replace it with,

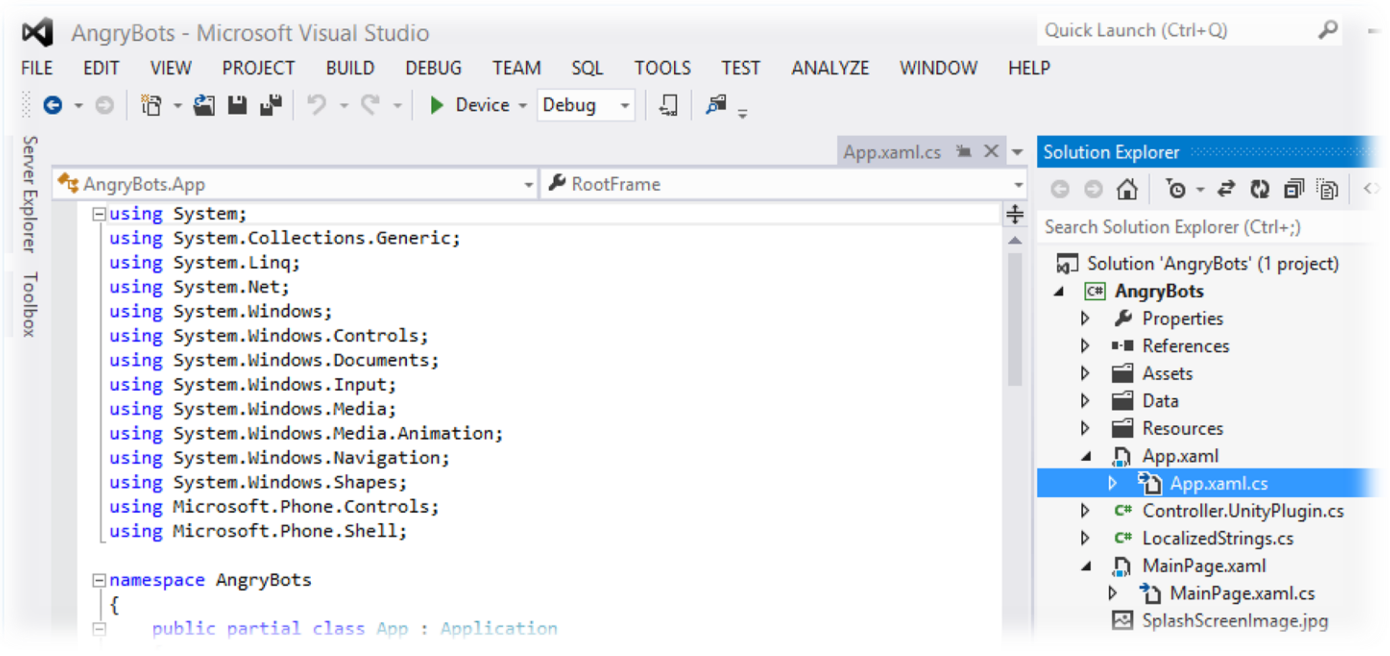
```
using Microsoft.Phone.Controls;
using Moga.Windows.Phone.Unity;
```

Update **Unity_Loaded** as shown below

```
private void Unity_Loaded()
{
        ControllerUnity.Connect(ControllerMode.Poll);
}
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.6.4 Update App.xaml



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace AngryBots
{
    public partial class App : Application
```

Open the file **App.xaml.cs** from the solution explorer; you will need to **click the dropdown arrow** on the left of **App.xaml** to see this file. The file will open in the Main pane, add the following;

Find the below,

```
using Microsoft.Phone.Shell;
```

And replace it with,

```
using Microsoft.Phone.Shell;
using Moga.Windows.Phone.Unity;
```

Update **Application_Activated** as shown below,

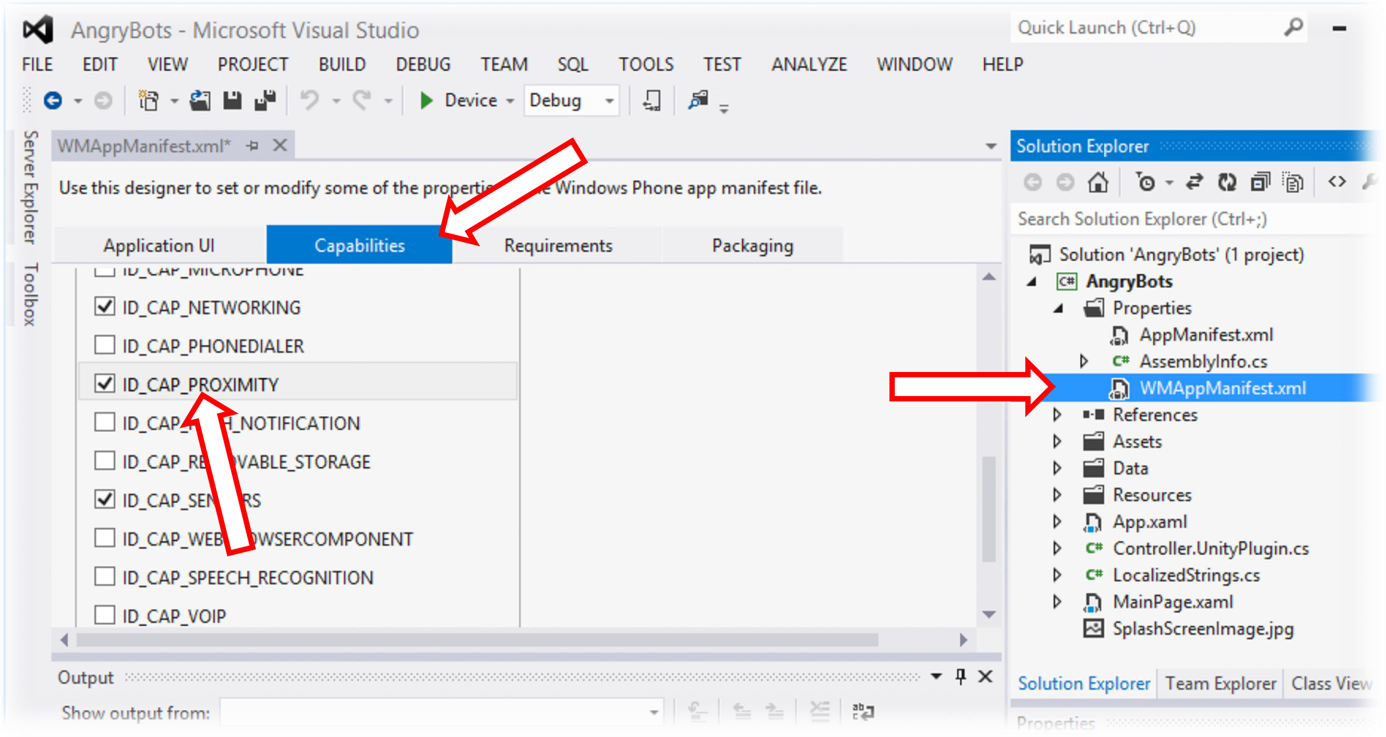
```
private void Application_Activated(object sender, ActivatedEventArgs e)
{
        ControllerUnity.Activated();
}
```

Update **Application_Deactivated** as shown below,

```
private void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
        UnityPlayer.UnityApp.Deactivate();
        ControllerUnity.Deactivated();
}
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.6.5 Update WPAppManifest.xml


AngryBots - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Device Debug
Server Explorer Toolbox
WMAppManifest.xml*
Use this designer to set or modify some of the properties in the Windows Phone app manifest file.
Application UI Capabilities Requirements Packaging
ID_CAP_MICROPHONE
☑ ID_CAP_NETWORKING
☐ ID_CAP_PHONEDIALER
☑ ID_CAP_PROXIMITY
☐ ID_CAP_PUSH_NOTIFICATION
☐ ID_CAP_REMOVABLE_STORAGE
☑ ID_CAP_SENSORS
☐ ID_CAP_WEBBROWSERCOMPONENT
☐ ID_CAP_SPEECH_RECOGNITION
☐ ID_CAP_VOIP
Output
Show output from:
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AngryBots' (1 project)
AngryBots
Properties
AppManifest.xml
AssemblyInfo.cs
WMAppManifest.xml
References
Assets
Data
Resources
App.xaml
Controller.UnityPlugin.cs
LocalizedStrings.cs
MainPage.xaml
SplashScreenImage.jpg
Solution Explorer Team Explorer Class View
Properties

- Finally double click the file **WPAppManifest.xml**, located in the Properties dropdown in the Solution Pane.
- Select the **Capabilities** tab
- Click enable **ID_CAP_PROXIMITY**.

Now click Build to experience Angry Bots with MOGA enabled controls.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 1.7 Tips

### 1.7.1 Retrieving the Controller Manager

You can check the Status of your MOGA Controller dynamically; first of all retrieve the MOGA Controller Manager game object during your Start or Awake methods,

```
private Moga_ControllerManager mogaControllerManager;

// Use this for initialization
void Start()
{
    mogaControllerObject = GameObject.Find("MogaControllerManager");

    if (mogaControllerObject != null)
    {
        // Check to see if the Moga_ControllerManager Script is attached.
        mogaControllerManager = mogaControllerObject.GetComponent<Moga_ControllerManager>();
    }
}
```

### 1.7.2 Determine Controller Connection

The below code returns true if the controller is connected, or false if it's disconnected.

```
if (mogaControllerManager != null)
{
    // MOGA Controller Manager Found
    Bool isConnected = mogaControllerManager.isControllerConnected();
}
```

### 1.7.3 Determine Controller Model

The below code returns true if the controller is a MOGA Pro, or false if it's a MOGA Pocket.

```
if (mogaControllerManager != null)
{
    // MOGA Controller Manager Found
    Bool isMogaPro = mogaControllerManager.isControllerMogaPro();
}
```

### 1.7.4 Enable GPS

You can enable GPS by opening **Moga_Input.cs** and uncommenting line 12,

```
//#define GPS_ENABLED
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

# Android

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.1 Android Plugin Essential Files

| | | |
|---|---|---|
| Standard Assets > | Scripts > | Moga_Controller.cs<br>Moga_ControllerManager.cs<br>Moga_Input.cs |
| Editor > | | Moga_Mapping.cs<br>Moga_Setup.cs<br>Texture_EditorHeader.psd |
| Plugins > | Android > | com.bda.controller.jar |

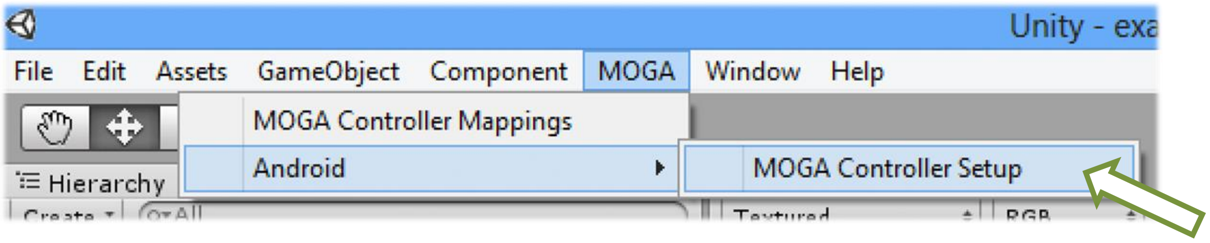
*Tip – Ensure the Texture_EditorHeader.psd is set as GUI for correct scaling/colour.*

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

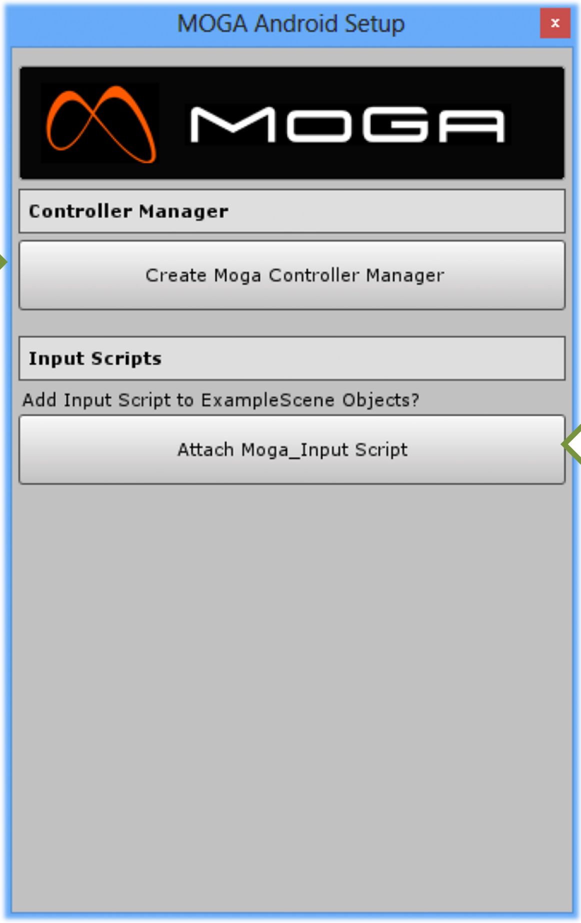
## 2.2 Creating the MOGA Controller Manager

The MOGA Controller Manager will allow us to manage all the Controllers in our application.

Click the **MOGA** Menu Item on the Unity Menu Bar and Select **MOGA Controller Setup** from the dropdown menu.



This will bring up the **MOGA Setup Editor Window** (shown below) which will allow us to manage our Controller Configuration,



**MOGA Controller Manager**
This will create a Game Object that manages the Controllers in our Scene.

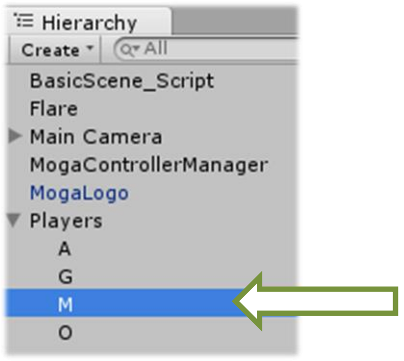
**MOGA Input Scripts**
Here we can attach Moga_Input Script to our desired Game Objects.

Click **Create MOGA Controller Manager,** this will create a **MogaControllerManage**r game object with the Moga_ControllerManager.cs script attached.

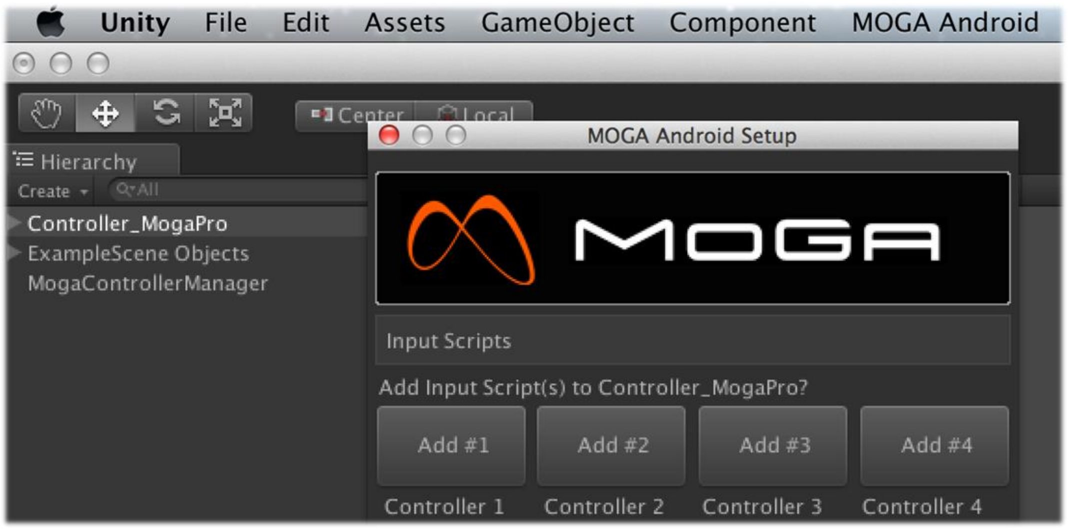
©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.3 Attaching the MOGA Input Script

Within the **Project Hierarchy**, select the Game Object that contains the Script for your controls.



The **MOGA Setup Editor Window** will change to reflect a Game Object has been selected, you will now see Four Additional buttons as shown below,



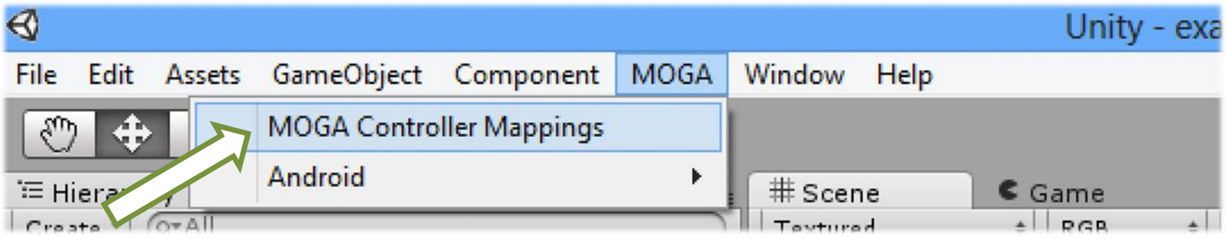
Click the desired Controller Number button to attach the **Moga_Input** Script for that specific controller to your Game Object.

You will notice the button changes to reflect your addition, re-clicking the button will remove the **Moga_Input** script from your Game Object.

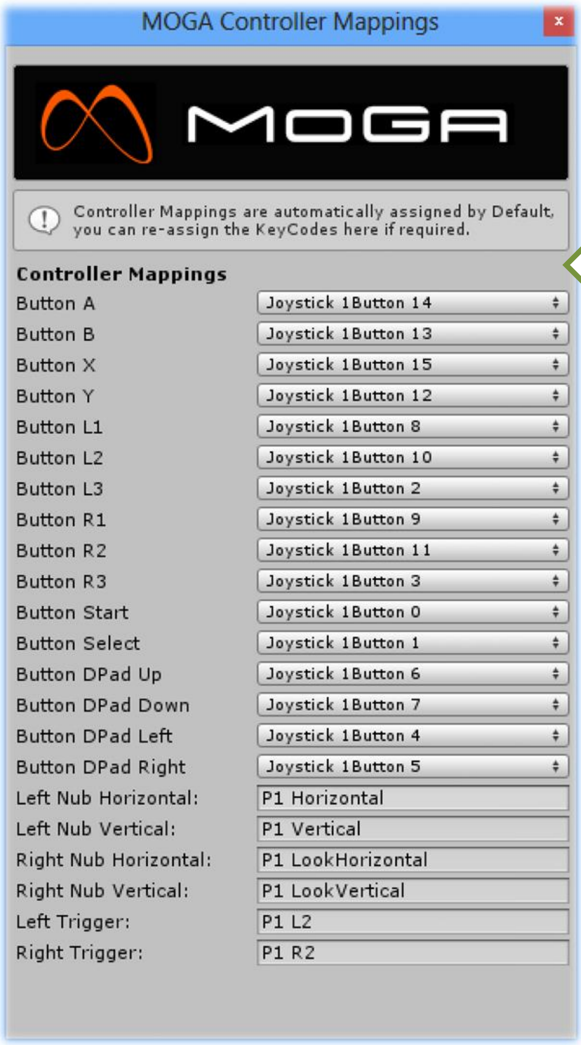
©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.4 Mapping the Controls (optional)

The final part of the MOGA Setup Editor Window allows you to map your Controller buttons and axis. You can omit this entirely and the controls will revert to the default mappings.



Selecting **MOGA Controller Mappings** from the **MOGA Android** Menu Entry will bring up the below Editor window,



Controller Mappings are automatically assigned by Default, you can re-assign the KeyCodes here if required.

**Controller Mappings**

| | |
|---|---|
| Button A | Joystick 1Button 14 |
| Button B | Joystick 1Button 13 |
| Button X | Joystick 1Button 15 |
| Button Y | Joystick 1Button 12 |
| Button L1 | Joystick 1Button 8 |
| Button L2 | Joystick 1Button 10 |
| Button L3 | Joystick 1Button 2 |
| Button R1 | Joystick 1Button 9 |
| Button R2 | Joystick 1Button 11 |
| Button R3 | Joystick 1Button 3 |
| Button Start | Joystick 1Button 0 |
| Button Select | Joystick 1Button 1 |
| Button DPad Up | Joystick 1Button 6 |
| Button DPad Down | Joystick 1Button 7 |
| Button DPad Left | Joystick 1Button 4 |
| Button DPad Right | Joystick 1Button 5 |
| Left Nub Horizontal: | P1 Horizontal |
| Left Nub Vertical: | P1 Vertical |
| Right Nub Horizontal: | P1 LookHorizontal |
| Right Nub Vertical: | P1 LookVertical |
| Left Trigger: | P1 L2 |
| Right Trigger: | P1 R2 |

**Default Mapping**
The KeyCode defaults are identical to the default iOS Controller Mappings unveiled in Unity 4.2.2. You can change them here if you wish.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.5 Managing the Controls

Open the Script that manages the Controls for your Unity Project. This should be attached to the same Game Object that you attached the Moga_Input Script to in Step 2.3.

At the very top of the Script just below your namespaces, add the following line;

**C# version**

```
using Input = Moga_Input;
```

**Javascript version**

```
private var Input : Moga_Input;
```

Next in your Start method you will need to register your controller, as show below;

**C# version**

```
void Start ()
{
    // Register MOGA Controller
    Input.RegisterMogaController();
}
```

**Javascript version**

```
void Start ()
{
    // Register Moga Controller
    Input = gameObject.GetComponent(Moga_Input);
    Input.RegisterMogaController();
}
```

Now you can use your Input.GetAxis and Input.GetKey, Input.GetKeyDown and Input.GetKeyUp as you would in Unity with the Controller Mappings you selected in Step 2.4.

The MOGA Scripts provided are in C#, when using them alongside JavaScript, it's important to note that JavaScript files are compiled before C# files.

To get the C# Script to compile first you must have placed the C# Scripts in the Standard Assets folder. Scripts placed within this location are compiled prior to any scripts present in other folders.

More information can be found here.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.6 Tips

### 2.6.1 Retrieving the Controller Manager

You can check the Status of your MOGA Controller dynamically; first of all retrieve the MOGA Controller Manager game object during your Start or Awake methods,

```
private Moga_ControllerManager mogaControllerManager;

// Use this for initialization
void Start()
{
    mogaControllerObject = GameObject.Find(“MogaControllerManager”);

    if (mogaControllerObject != null)
    {
        // Check to see if the Moga_ControllerManager Script is attached.
        mogaControllerManager = mogaControllerObject.GetComponent<Moga_ControllerManager>();
    }
}
```

### 2.6.2 Determine Controller Connection

The below code returns true if the controller is connected, or false if it’s disconnected.

```
if (mogaControllerManager != null)
{
    // MOGA Controller Manager Found
    Bool isConnected = mogaControllerManager.isControllerConnected();
}
```

### 2.6.3 Determine Controller Model

The below code returns true if the controller is a MOGA Pro, or false if it’s a MOGA Pocket.

```
if (mogaControllerManager != null)
{
    // MOGA Controller Manager Found
    Bool isMogaPro = mogaControllerManager.isControllerMogaPro();
}
```

### 2.6.4 Enable GPS

You can enable GPS by opening **Moga_Input.cs** and uncommenting line 12,

```
//#define GPS_ENABLED
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

## 2.7 Adding MOGA Android/WP8 Support to Angry Bots

### 2.7.1 Importing the MOGA Plugin

Download the MOGA Plugin from the Unity Asset Store, for this tutorial we don't need to import the bundled example scenes, so import only the below files,

**Plugin Requirements**

| | |
|---|---|
| Standard Assets > | Scripts |
| Editor > | Moga_Mapping.cs<br>Moga_Setup.cs<br>PostProcessBuildPlayer_Moga.cs<br>Texture_EditorHeader.psd |
| Plugins > | winphone8<br>Android |

*Tip – Ensure the Texture_EditorHeader.psd is set as GUI for correct scaling/colour.*

**Requirements**

- AngryBots v4.0 (bundled with Unity 4.0)

**Android Requirements**

- Android SDK
- Java SDK 1.6+

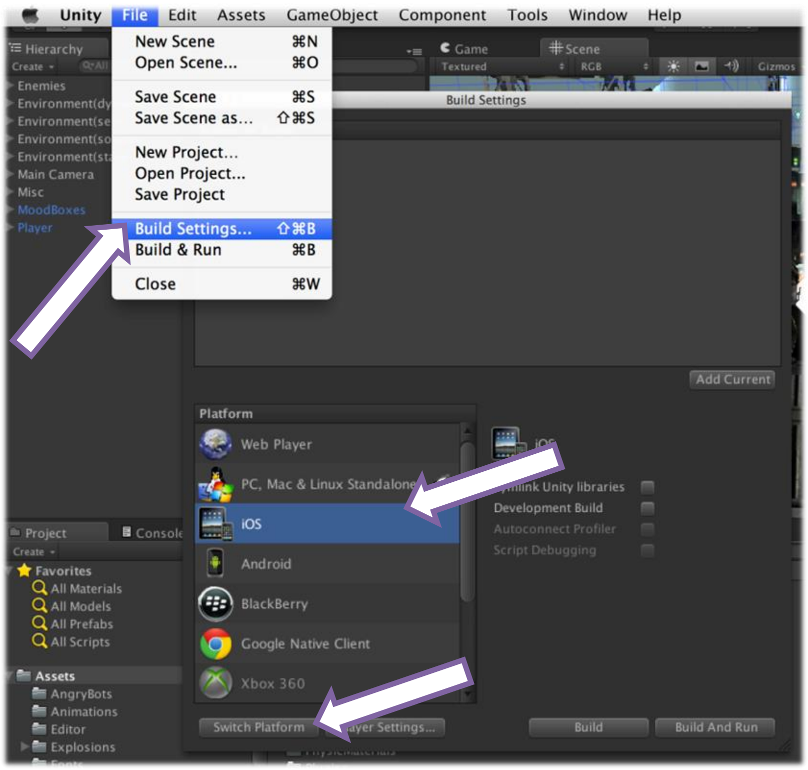
**WP8 Requirements**

- Visual Studio

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

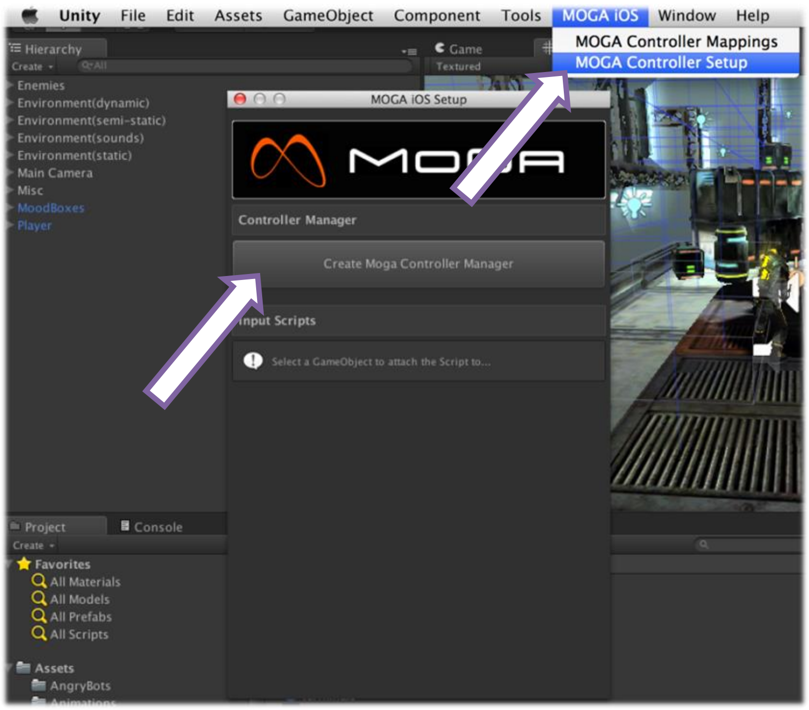
### 2.7.2 Change Project Build to Android

- Load the Angrybots project and open the included Angrybots.unity scene.
- Click **File > Build Settings** or alternatively press **Shift + Command + B** to open the Build Settings window.
- In the **Platform section**, choose **Android** and press the **Switch Platform** button to begin the projects conversion.



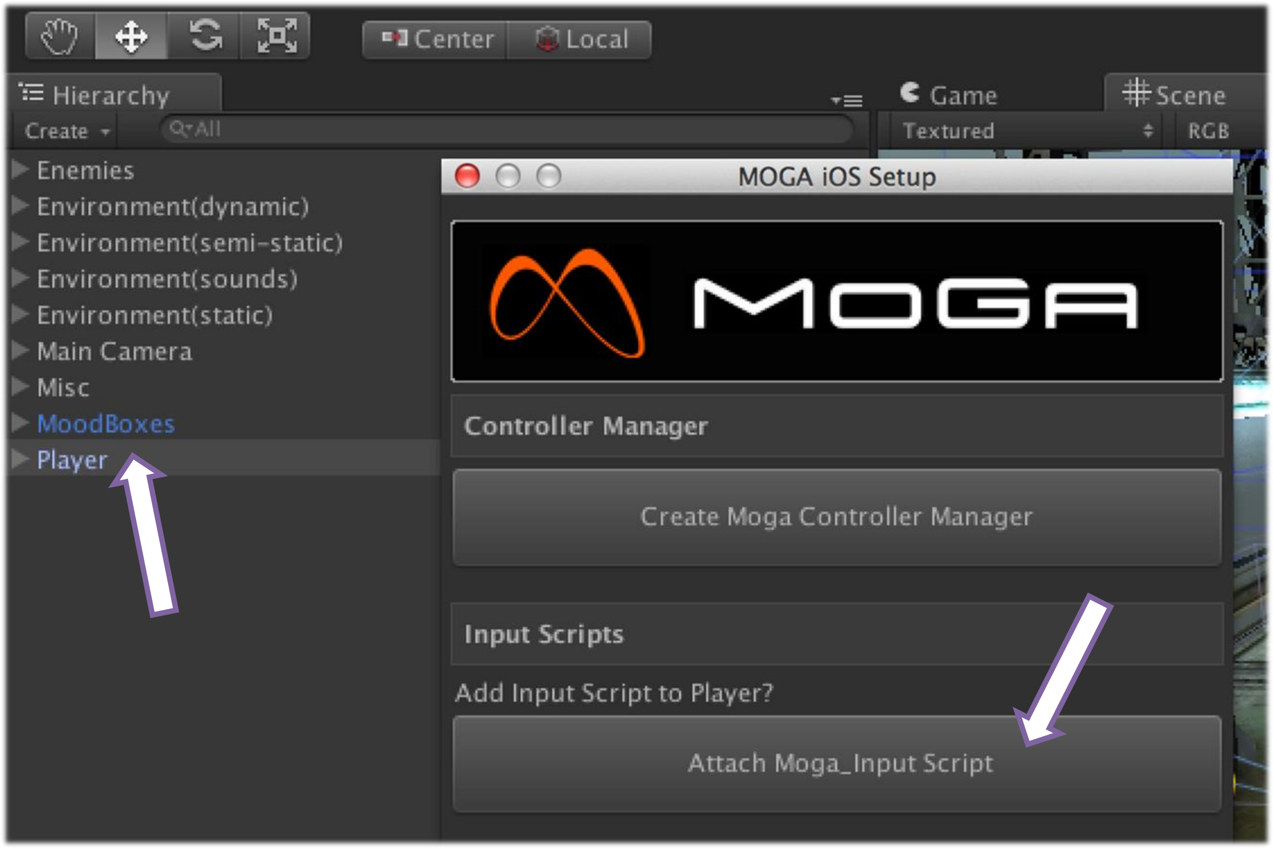
### 2.7.3 Open the MOGA Controller Setup Window

- Click the **MOGA Android** Menu from the menu bar, if the option is not visible, click on the menu bar to make it appear.
- Choose **MOGA Controller Setup** from the drop down.
- Click **Create Moga Controller Manager**.


Unity File Edit Assets GameObject Component Tools MOGA IOS Window Help
MOGA Controller Mappings
MOGA Controller Setup
MOGA iOS Setup
MOGA
Controller Manager
Create Moga Controller Manager
Input Scripts
Select a GameObject to attach the Script to...

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

### 2.7.4 Attach the MOGA_Input Script to the Player GameObject



- Select the **Player** GameObject from the Projects **Hierarchy**.
- You will notice the MOGA Input Scripts becomes available in the MOGA Setup window.
- Select **Attach Moga_Input Script** to attach a MOGA_Input Script to the Player GameObject.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

### 2.7.5 Edit Scripts > Movement > PlayerMoveController.js

Find the below on **line 40**,

```
private var screenMovementRight : Vector3;
```

And replace it with,

```
private var screenMovementRight : Vector3;

#if  UNITY_ANDROID || UNITY_WP8
private var Input : Moga_Input;
#endif
```

Find the below on **line 91**,

```
function Start () {
```

And replace it with,

```
function Start () {

   // Register Moga Controller
   #if UNITY_WP8 || UNITY_ANDROID
   Input = gameObject.GetComponent(Moga_Input);
   Input.RegisterMogaController();
   #endif
```

Find the below on **line 130 – 135**,

```
// HANDLE CHARACTER MOVEMENT DIRECTION
#if UNITY_IPHONE || UNITY_ANDROID || UNITY_WP8 || UNITY_BLACKBERRY
   motor.movementDirection = joystickLeft.position.x * screenMovementRight + joystickLeft.position.y * screenMovementForward;
#else
   motor.movementDirection = Input.GetAxis ("Horizontal") * screenMovementRight + Input.GetAxis ("Vertical") * screenMovementForward;
#endif
```

And replace it with,

```
// HANDLE CHARACTER MOVEMENT DIRECTION
#if UNITY_IPHONE || UNITY_BLACKBERRY
motor.movementDirection = joystickLeft.position.x * screenMovementRight + joystickLeft.position.y * screenMovementForward;
#elif UNITY_WP8 || UNITY_ANDROID
var moveAxisX : float = Input.GetAxis("Horizontal");
var moveAxisY : float = Input.GetAxis("Vertical");

if(moveAxisX != 0 || moveAxisY != 0)  {
        motor.movementDirection = moveAxisX * screenMovementRight + moveAxisY * screenMovementForward;
} else {
        motor.movementDirection = joystickLeft.position.x * screenMovementRight + joystickLeft.position.y * screenMovementForward;
}
#endif
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

### 2.7.6 Edit Scripts > Movement > PlayerMoveController.js (continued)

Find the below on **line 168 - 175**,

```
#if UNITY_IPHONE || UNITY_ANDROID || UNITY_WP8 || UNITY_BLACKBERRY

// On mobiles, use the thumb stick and convert it into screen movement space
motor.facingDirection = joystickRight.position.x * screenMovementRight + joystickRight.position.y * screenMovementForward;

cameraAdjustmentVector = motor.facingDirection;

#else
```

And replace it with,

```
#if UNITY_IPHONE || UNITY_BLACKBERRY

// On mobiles, use the thumb stick and convert it into screen movement space
motor.facingDirection = joystickRight.position.x * screenMovementRight + joystickRight.position.y * screenMovementForward;

cameraAdjustmentVector = motor.facingDirection;

#elif UNITY_WP8 || UNITY_ANDROID

var axisX : float = Input.GetAxis("LookHorizontal");
var axisY : float = Input.GetAxis("LookVertical");

if(axisX != 0 || axisY != 0)
{
	motor.facingDirection = axisX * screenMovementRight + axisY * screenMovementForward;
}
else
{
	motor.facingDirection = joystickRight.position.x * screenMovementRight + joystickRight.position.y * screenMovementForward;
}

cameraAdjustmentVector = motor.facingDirection;

#else
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

### 2.7.7 Edit Scripts > Modules > TriggerOnMouseOrJoystick.js

Find the below on **line 6**,

```
private var state : boolean = false;
```

And replace it with,

```
private var state : boolean = false;

#if UNITY_WP8 || UNITY_ANDROID
private var Input : Moga_Input;
#endif
```

Find the below on **line 23-32**,

```
#if UNITY_IPHONE || UNITY_ANDROID || UNITY_WP8 || UNITY_BLACKBERRY
if (state == false && joysticks[0].tapCount > 0)
{
        mouseDownSignals.SendSignals (this);
        state = true;
}
else if (state == true && joysticks[0].tapCount > 0)
{
        mouseUpSignals.SendSignals (this);
        state = false;
}
#else
```

And replace it with,

```
#if UNITY_IPHONE || UNITY_BLACKBERRY
if (state == false && joysticks[0].tapCount > 0)  {
        mouseDownSignals.SendSignals (this);
        state = true;

} else if (state == true && joysticks[0].tapCount > 0)  {
        mouseUpSignals.SendSignals (this);
        state = false;
}
#elif UNITY_WP8 || UNITY_ANDROID
if (state == false && (joysticks[0].tapCount > 0 || Input.GetKeyDown(KeyCode.Joystick1Button11))) {
        mouseDownSignals.SendSignals (this);
        state = true;

} else if (!Input.GetKey(KeyCode.Joystick1Button11)) {

        if(state == true && joysticks[0].tapCount == 0)  {
                mouseUpSignals.SendSignals (this);
                state = false;
        }
}
#else
```

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.

#### 2.7.8 Build and Run

The MOGA Plugin features a one-click automated build process whereby the Visual Studio (WP8) build will automatically be automatically updated with the necessary code, libraries and references.

Tip - If you would like to manually build your Visual Studio project you can delete the **PostProcessBuildPlayer_Moga.cs** script from your Editor folder and follow the steps in 1.6 for Visual Studio.

To build select **File > Build and Run** from the Unity Menu.

If you want to build for WP8, simply bring up the Build Settings window again and switch to the desired platform before choosing Build and Run.

©2013 Bensussen Deutsch and Associates, Inc. All rights reserved.